Movie Ticket Booking System

A course project report

By

Haripreeth Dwarakanath Avarur [RA2011003010011]
Kartik Paliwal [RA2011003010013]
Aditi [RA2011003010004]

Under the guidance of

Dr. Muruganantham B.

(Associate Professor, Department of Computing Technologies)

In partial fulfillment for the course of

18CSC303J – Database Management System

in the department of Computing Technologies,
School of Computing,
Faculty of Engineering and Technology,
For the degree of

Bachelor of Technology in Computer Science and Engineering

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
Kattankulathur, Chengalpattu Taluk, Kanchipuram District, Tamil Nadu, India - 603203

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "Movie Ticket Booking System" is the bona fide work of "Haripreeth Dwarakanath Avarur", "Kartik Paliwal" and "Aditi", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Muruganantham B.
Associate Professor,
Department of Computing Technologies,
SRM Institute of Science and Technology

TABLE OF CONTENTS

ABSTRACT:

The Movie Ticket Booking System is a mini project that is designed to streamline the process of booking tickets for movies online. This system enables users to view the latest movie releases, check the availability of seats, and book tickets online. The project consists of two parts: a backend system that stores and manages movie data, and a frontend user interface that allows users to interact with the system. The frontend is designed using the Tkinter library in Python, while the backend uses SQLite for data storage and management. The system provides features such as adding, updating, and deleting movie records, searching for movies, and displaying movie details. Overall, the Movie Ticket Booking System aims to provide a convenient and efficient platform for users to book movie tickets online.

INTRODUCTION:

The Movie Ticket Booking System is a mini project that utilizes the concepts of database management and GUI programming to create an efficient and user-friendly system for booking movie tickets online. The project has been designed to address the challenges faced by moviegoers in purchasing tickets, such as long queues, limited availability of tickets, and inconvenient payment options. By implementing this system, movie enthusiasts can easily book tickets online, view movie schedules and timings, and choose their preferred seats in the cinema hall.

This project is an excellent example of how database management systems (DBMS) can be used to create an efficient and effective system for managing data related to movies, such as movie titles, release dates, cast, crew, budget, duration, and ratings. The system is built using Python and the Tkinter library for GUI programming, making it easy to use and understand for both developers and end-users.

The purpose of this report is to provide an overview of the Movie Ticket Booking System, including its objectives, problem statement, approach, proposed solution, scope, and architecture. We will also discuss how our solution differs from existing systems and present the use case and architecture diagrams. Overall, this project aims to enhance the user experience of booking movie tickets and simplify the process of managing movie-related data using DBMS.

PROBLEM STATEMENT:

The Movie Ticket Booking System aims to provide an efficient and user-friendly way for customers to book movie tickets online. The traditional method of purchasing movie tickets involves standing in long queues at the ticket counter, which can be time-consuming and inconvenient. With this system, customers can easily browse through the list of available movies, select the desired show timings, and book their tickets online from the comfort of their homes. The system also allows customers to view details about the movie such as its cast, director, release date, duration, and rating. The primary goal of this system is to make the process of booking movie tickets hassle-free and convenient for customers.

OBJECTIVES:

The objective of the "Movie Ticket Booking System" DBMS mini project is to develop a database management system that can manage movie data, such as movie name, release date, director, cast, budget, duration, and rating. The system should provide functionalities for adding, deleting, updating, and searching movie data. Additionally, the system should allow users to book movie tickets online and provide a platform for movie theatre managers to manage movie screenings, such as scheduling and seat availability. The system should also ensure data security and user privacy by providing authentication and authorization mechanisms. Overall, the system should provide an efficient and user-friendly interface to manage movie data and facilitate online movie ticket booking.

APPROACH:

The primary approach followed in the project is a combination of frontend and backend development.

The frontend of the system is built using the Tkinter module in Python, which is a widely-used graphical user interface (GUI) toolkit for Python. Tkinter is a powerful and easy-to-use GUI toolkit that is built on top of the Tk GUI toolkit. It provides a variety of widgets that can be used to create rich and interactive user interfaces, such as buttons, labels, text boxes, and more.

The backend of the system is built using Python and SQLite. SQLite is a lightweight, fast, and reliable relational database management system that is widely used in small-scale applications. In this project, we use SQLite to store and manage the data related to movies, such as movie name, ID, release date, director, cast, budget, duration, and rating.

To develop the project, we followed a systematic approach that involved several steps, such as requirement gathering, design, implementation, testing, and deployment. We started by gathering the requirements for the project, which included the features and functionalities that the system should have.

Next, we designed the system by creating a detailed blueprint of the frontend and backend components of the system. This included designing the user interface, creating the database schema, and defining the functions that would be used to interact with the database.

After designing the system, we implemented it by writing the code for both the frontend and backend components. We used Python for both the frontend and backend development, which helped us to maintain consistency and reduce the complexity of the project.

Once the implementation was completed, we tested the system thoroughly to ensure that it was working as expected. We performed both manual and automated testing, which helped us to identify and fix any issues or bugs in the system.

Finally, we deployed the system to a local machine, where it can be accessed by users to book movie tickets online.

Overall, the approach used to develop the Movie Ticket Booking System involved a combination of frontend and backend development, a systematic development process, and a focus on quality assurance and testing. This approach helped us to build a robust and reliable system that is easy to use and meets the requirements of its users.

PROPOSED SOLUTION:

The proposed solution is a software system that provides an online booking system to customers and helps the cinema to manage movie details. The system is designed to be user-friendly, secure, and easy to use.

FRONTEND CODE:

```
#Frontend

from tkinter import *
import tkinter.messagebox
import MiniProject_Backend

class Movie:
    def __init__(self, root):
        self.root=root
        self.root.title("Movie Ticket Booking System")
        self.root.geometry("1350x750+0+0")
        self.root.config(bg="blue")

        Movie_Name=StringVar()
        Movie_ID=StringVar()
        Release_Date=StringVar()
        Director=StringVar()
        Cast=StringVar()
        Budget=StringVar()
        Duration=StringVar()
        Rating=StringVar()

        #Fuctions
        def iExit():
            iExit=tkinter.messagebox.askyesno("Movie Ticket Booking System", "Are you sure???")
            if iExit>0:
                root.destroy()
            return

        def clcdata():
            self.txtMovie_ID.delete(0,END)
            self.txtMovie_Name.delete(0,END)
            self.txtRelease_Date.delete(0,END)
            self.txtDirector.delete(0,END)
            self.txtCast.delete(0,END)
            self.txtBudget.delete(0,END)
            self.txtRating.delete(0,END)
            self.txtDuration.delete(0,END)

        def adddata():
            if(len(Movie_ID.get())!=0):
                MiniProject_Backend.add_movie_record(Movie_ID.get(),Movie_Name.get(),Release_Date.get(),Director.get(),Cast.get(),Budget.get(),Duration.get(),Rating.get())
                MovieList.delete(0,END)
                MovieList.insert(END,(Movie_ID.get(),Movie_Name.get(),Release_Date.get(),Director.get(),Cast.get(),Budget.get(),Duration.get(),Rating.get()))

        def disdata():
            MovieList.delete(0,END)
            for row in MiniProject_Backend.ViewMovieData():
                MovieList.insert(END, row, str(""))

        def movierec(event):
            global sd
            searchmovie=MovieList.curselection()[0]
            sd=MovieList.get(searchmovie)
```

```
            self.txtMovie_ID.delete(0,END)
            self.txtMovie_ID.insert(END,sd[1])
            self.txtMovie_Name.delete(0,END)
            self.txtMovie_Name.insert(END,sd[2])
            self.txtRelease_Date.delete(0,END)
            self.txtRelease_Date.insert(END,sd[3])
            self.txtDirector.delete(0,END)
            self.txtDirector.insert(END,sd[4])
            self.txtCast.delete(0,END)
            self.txtCast.insert(END,sd[5])
            self.txtBudget.delete(0,END)
            self.txtBudget.insert(END,sd[6])
            self.txtDuration.delete(0,END)
            self.txtDuration.insert(END,sd[7])
            self.txtRating.delete(0,END)
            self.txtRating.insert(END,sd[8])

        def deldata():
            if(len(Movie_ID.get())!=0):
                MiniProject_Backend.DeleteMovieRec(sd[0])
                clcdata()
                disdata()

        def searchdb():
            MovieList.delete(0,END)
            for row in MiniProject_Backend.SearchMovieData(Movie_ID.get(),Movie_Name.get(),Release_Date.get(),Director.get(),Cast.get(),Budget.get(),Duration.get(),Rating.get()):
                MovieList.insert(END, row, str(""))

        def updata():
            if(len(Movie_ID.get())!=0):
                MiniProject_Backend.DeleteMovieRec(sd[0])
            if(len(Movie_ID.get())!=0):
                MiniProject_Backend.add_movie_record(Movie_ID.get(),Movie_Name.get(),Release_Date.get(),Director.get(),Cast.get(),Budget.get(),Duration.get(),Rating.get())
                MovieList.delete(0,END)
                MovieList.insert(END,(Movie_ID.get(),Movie_Name.get(),Release_Date.get(),Director.get(),Cast.get(),Budget.get(),Duration.get(),Rating.get()))

        #Frames
        MainFrame=Frame(self.root, bg="blue")
        MainFrame.grid()

        TFrame=Frame(MainFrame, bd=5, padx=54, pady=8, bg="blue", relief=RIDGE)
        TFrame.pack(side=TOP)

        self.TFrame=Label(TFrame, font=('Arial', 51, 'bold'), text="ONLINE MOVIE TICKET BOOKING SYSTEM", bg="blue", fg="orange")
        self.TFrame.grid()

        BFrame=Frame(MainFrame, bd=2, width=1350, height=70, padx=18, pady=10, bg="blue", relief=RIDGE)
        BFrame.pack(side=BOTTOM)

        DFrame=Frame(MainFrame, bd=2, width=1300, height=400, padx=20, pady=20, bg="blue", relief=RIDGE)
        DFrame.pack(side=BOTTOM)

        DFrameL=LabelFrame(DFrame, bd=2, width=1000, height=600, padx=20, bg="blue", relief=RIDGE, font=('Arial', 20, 'bold'), text="Movie Info_\n", fg="white")
        DFrameL.pack(side=LEFT)
```

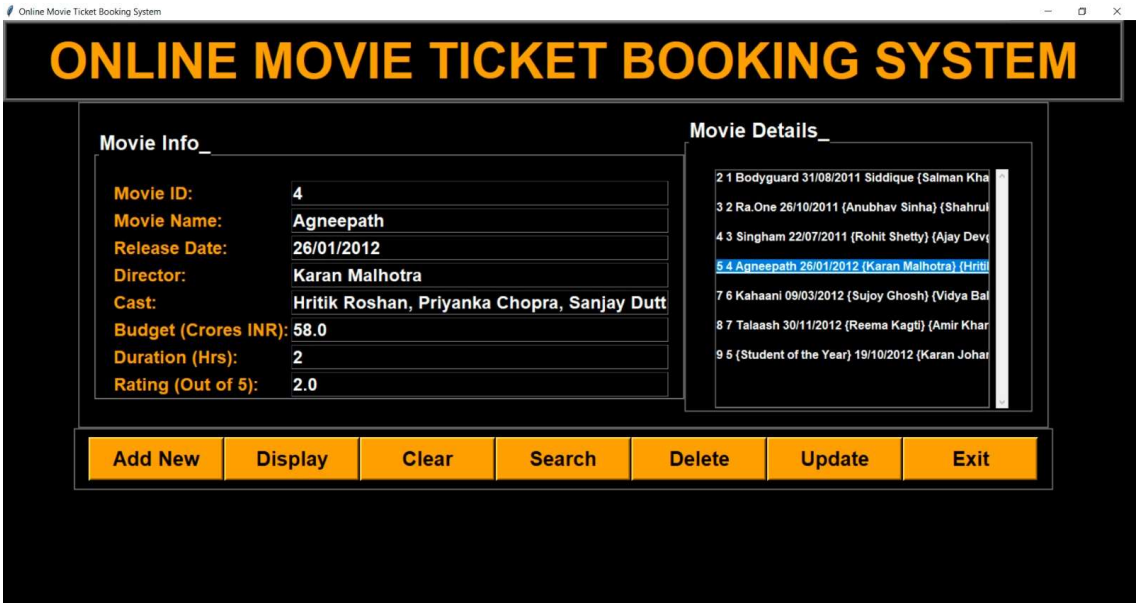
```
    DFrameR=LabelFrame(DFrame, bd=2, width=450, height=300, padx=31, pady=3, bg="blue", relief=RIDGE, font=('Arial', 20, 'bold'), text="Movie Details_\n", fg="white")
    DFrameR.pack(side=RIGHT)

    #Labels & Entry Box

    self.lblMovie_ID=Label(DFrameL, font=('Arial', 18, 'bold'), text="Movie ID:", padx=2, pady=2, bg="blue", fg="orange")
    self.lblMovie_ID.grid(row=0, column=0, sticky=W)
    self.txtMovie_ID=Entry(DFrameL, font=('Arial', 18, 'bold'), textvariable=Movie_ID, width=39, bg="blue", fg="white")
    self.txtMovie_ID.grid(row=0, column=1)

    self.lblMovie_Name=Label(DFrameL, font=('Arial', 18, 'bold'), text="Movie Name:", padx=2, pady=2, bg="blue", fg="orange")
    self.lblMovie_Name.grid(row=1, column=0, sticky=W)
    self.txtMovie_Name=Entry(DFrameL, font=('Arial', 18, 'bold'), textvariable=Movie_Name, width=39, bg="blue", fg="white")
    self.txtMovie_Name.grid(row=1, column=1)

    self.lblRelease_Date=Label(DFrameL, font=('Arial', 18, 'bold'), text="Release Date:", padx=2, pady=2, bg="blue", fg="orange")
    self.lblRelease_Date.grid(row=2, column=0, sticky=W)
    self.txtRelease_Date=Entry(DFrameL, font=('Arial', 18, 'bold'), textvariable=Release_Date, width=39, bg="blue", fg="white")
    self.txtRelease_Date.grid(row=2, column=1)

    self.lblDirector=Label(DFrameL, font=('Arial', 18, 'bold'), text="Director:", padx=2, pady=2, bg="blue", fg="orange")
    self.lblDirector.grid(row=3, column=0, sticky=W)
    self.txtDirector=Entry(DFrameL, font=('Arial', 18, 'bold'), textvariable=Director, width=39, bg="blue", fg="white")
    self.txtDirector.grid(row=3, column=1)

    self.lblCast=Label(DFrameL, font=('Arial', 18, 'bold'), text="Cast:", padx=2, pady=2, bg="blue", fg="orange")
    self.lblCast.grid(row=4, column=0, sticky=W)
    self.txtCast=Entry(DFrameL, font=('Arial', 18, 'bold'), textvariable=Cast, width=39, bg="blue", fg="white")
    self.txtCast.grid(row=4, column=1)

    self.lblBudget=Label(DFrameL, font=('Arial', 18, 'bold'), text="Budget (Crores INR):", padx=2, pady=2, bg="blue", fg="orange")
    self.lblBudget.grid(row=5, column=0, sticky=W)
    self.txtBudget=Entry(DFrameL, font=('Arial', 18, 'bold'), textvariable=Budget, width=39, bg="blue", fg="white")
    self.txtBudget.grid(row=5, column=1)

    self.lblDuration=Label(DFrameL, font=('Arial', 18, 'bold'), text="Duration (Hrs):", padx=2, pady=2, bg="blue", fg="orange")
    self.lblDuration.grid(row=6, column=0, sticky=W)
    self.txtDuration=Entry(DFrameL, font=('Arial', 18, 'bold'), textvariable=Duration, width=39, bg="blue", fg="white")
    self.txtDuration.grid(row=6, column=1)

    self.lblRating=Label(DFrameL, font=('Arial', 18, 'bold'), text="Rating (Out of 5):", padx=2, pady=2, bg="blue", fg="orange")
    self.lblRating.grid(row=7, column=0, sticky=W)
    self.txtRating=Entry(DFrameL, font=('Arial', 18, 'bold'), textvariable=Rating, width=39, bg="blue", fg="white")
    self.txtRating.grid(row=7, column=1)

    #ListBox & ScrollBar
    sb=Scrollbar(DFrameR)
    sb.grid(row=0, column=1, sticky='ns')

    MovieList=Listbox(DFrameR, width=41, height=16, font=('Arial', 12, 'bold'), bg="blue", fg="white", yscrollcommand=sb.set)
    MovieList.bind('<<ListboxSelect>>', movierec)
    MovieList.grid(row=0, column=0, padx=8)
    sb.config(command=MovieList.yview)
```

```
        #Buttons
        self.btnadd=Button(BFrame, text="Add New", font=('Arial', 20, 'bold'), width=10, height=1, bd=4, bg="orange", command=adddata)
        self.btnadd.grid(row=0, column=0)

        self.btndis=Button(BFrame, text="Display", font=('Arial', 20, 'bold'), width=10, height=1, bd=4, bg="orange", command=disdata)
        self.btndis.grid(row=0, column=1)

        self.btnclc=Button(BFrame, text="Clear", font=('Arial', 20, 'bold'), width=10, height=1, bd=4, bg="orange", command=clcdata)
        self.btnclc.grid(row=0, column=2)

        self.btnse=Button(BFrame, text="Search", font=('Arial', 20, 'bold'), width=10, height=1, bd=4, bg="orange", command=searchdb)
        self.btnse.grid(row=0, column=3)

        self.btndel=Button(BFrame, text="Delete", font=('Arial', 20, 'bold'), width=10, height=1, bd=4, bg="orange", command=deldata)
        self.btndel.grid(row=0, column=4)

        self.btnup=Button(BFrame, text="Update", font=('Arial', 20, 'bold'), width=10, height=1, bd=4, bg="orange", command=updata)
        self.btnup.grid(row=0, column=5)

        self.btnx=Button(BFrame, text="Exit", font=('Arial', 20, 'bold'), width=10, height=1, bd=4, bg="orange", command=iExit)
        self.btnx.grid(row=0, column=6)


if __name__=='__main__':
    root=Tk()
    datbase=Movie(root)
    root.mainloop()
```

BACKEND CODE:

```python
# backend.py
import sqlite3
# Create the movies table
def create_movies_table():
    with sqlite3.connect("movie1.db") as con:
        cur = con.cursor()
        cur.execute("""CREATE TABLE IF NOT EXISTS movies (
                        id INTEGER PRIMARY KEY,
                        movie_id TEXT,
                        movie_name TEXT,
                        release_date TEXT,
                        director TEXT,
                        cast TEXT,
                        budget TEXT,
                        duration TEXT,
                        rating TEXT
                    )""")
        con.commit()

# Add a new movie record to the movies table
def add_movie_record(movie_id, movie_name, release_date, director, cast, budget, duration, rating):
    with sqlite3.connect("movie1.db") as con:
        cur = con.cursor()
        cur.execute("""INSERT INTO movies (movie_id, movie_name, release_date, director, cast, budget, duration, rating)
                        VALUES (?, ?, ?, ?, ?, ?, ?, ?)""",
                    (movie_id, movie_name, release_date, director, cast, budget, duration, rating))
        con.commit()

# View all the movie records in the movies table
def view_movie_records():
    with sqlite3.connect("movie1.db") as con:
        cur = con.cursor()
        cur.execute("SELECT * FROM movies")
        rows = cur.fetchall()
        return rows

# Delete a movie record from the movies table
def delete_movie_record(movie_id):
    with sqlite3.connect("movie1.db") as con:
        cur = con.cursor()
        cur.execute("DELETE FROM movies WHERE movie_id=?", (movie_id,))
        con.commit()

# Search for a movie record in the movies table
def search_movie_records(movie_id="", movie_name="", release_date="", director="", cast="", budget="", duration="", rating=""):
    with sqlite3.connect("movie1.db") as con:
        cur = con.cursor()
        cur.execute("""SELECT * FROM movies
                        WHERE movie_id=? OR movie_name=? OR release_date=? OR director=?
                        OR cast=? OR budget=? OR duration=? OR rating=?""",
                    (movie_id, movie_name, release_date, director, cast, budget, duration, rating))
        rows = cur.fetchall()
        return rows

# Update a movie record in the movies table
def update_movie_record(movie_id, movie_name="", release_date="", director="", cast="", budget="", duration="", rating=""):
    with sqlite3.connect("movie1.db") as con:
        cur = con.cursor()
        cur.execute("""UPDATE movies
                        SET movie_name=?, release_date=?, director=?, cast=?, budget=?, duration=?, rating=?
                        WHERE movie_id=?""",
                    (movie_name, release_date, director, cast, budget, duration, rating, movie_id))
        con.commit()
```

OUTPUT:



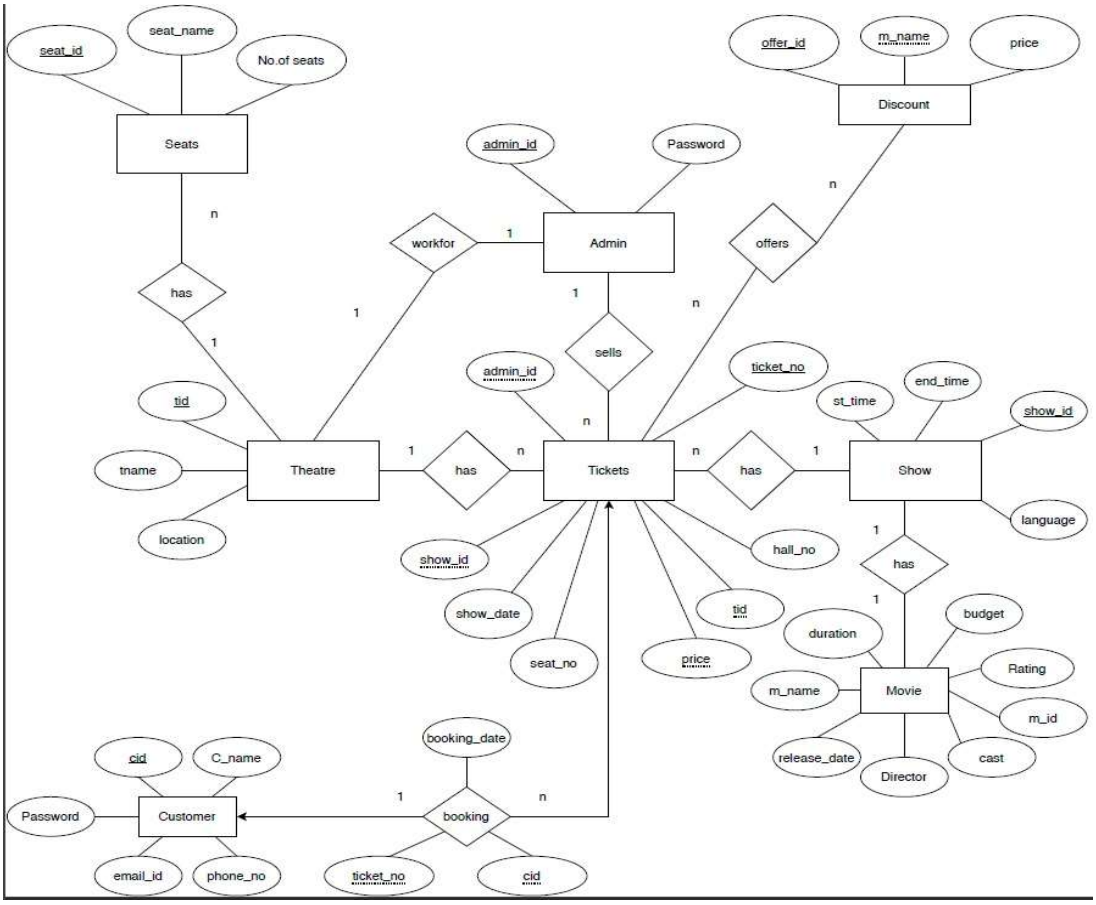
SCOPE:

The scope of the project is limited to managing movie details and providing an online booking system to customers. The system is designed to be scalable and can be extended to include more features.

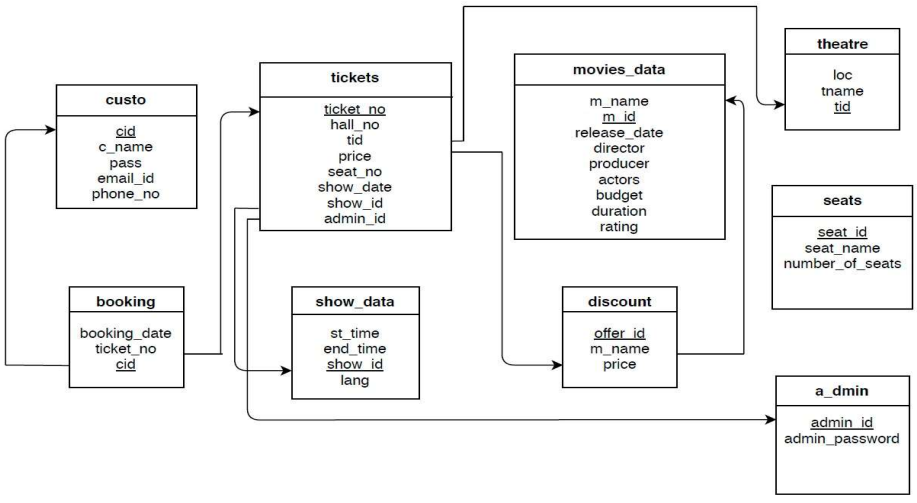
HOW OUR SOLUTION DIFFERS:

Our solution differs from other solutions as it is designed to be user-friendly, secure, and easy to use. It provides an online booking system to customers, which makes it convenient for them to book movie tickets. It also helps the cinema to manage movie details efficiently.

ENTITY – RELATIONSHIP DIAGRAM:



SCHEMA:



CONCLUSION:

The Movie Ticket Booking System is a mini project based on the concept of database management systems. It provides a user-friendly interface for booking movie tickets online. The system includes both the front-end interface designed using the Tkinter library in Python and the back-end database management system. The front-end interface allows the user to input details of movies, view movie details, search for movies, update movie information, and delete movie information. The back-end database management system stores and retrieves movie details efficiently using SQL queries. The project aims to make the process of booking movie tickets more convenient and accessible to users. Overall, the Movie Ticket Booking System provides a robust and efficient solution for managing movie ticket bookings online.

REFERENCES:

- https://docs.python.org/
- https://docs.python.org/3/library/tk.html
- https://www.sqlite.org/docs.html
- https://www.w3schools.com/sql/
- https://www.youtube.com/watch?v=zbMHLJ0dY4w
- https://www.youtube.com/watch?v=HXV3zeQKqGY