MOVIE RESERVATION MANAGEMENT SYSTEM

A MINI PROJECT REPORT

SUBMITTED BY

AVULA SNEYA DRITI	**221701008**
LIVIA MARY SEBASTIAN	**221701033**
MANJUSHREE M	**221701036**
SNEHA S	**221701055**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND DESIGN ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE
THANDALAM
CHENNAI – 602105

ANNA UNIVERSITY: CHENNAI 600625

BONAFIDE CERTIFICATE

Certified that this project report **“MOVIE RESERVATION MANAGEMENT SYSTEM”** is the Bonafide work of **“AVULA SNEYA DRITI(221701008), LIVIA MARY SEBASTIAN(221701033), MANJUSHREE M (221701036), SNEHA S (200701312)**” who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Mr. UmaMaheswar Rao.,	**Mr.Vijaykumar M.Tech.,**
Professor and Head,	Asst. Professor (SS),
Computer Science and Design Engineering,	Computer Science and Engineering,
Rajalakshmi Engineering College,	Rajalakshmi Engineering College,
Thandalam, Chennai – 602105.	Thandalam, Chennai – 602105.

EXTERNAL EXAMINER **INTERNAL EXAMINER**

ACKNOWLEGEMENT

We are highly obliged in taking the opportunity to thank our Chairman **Mr. S. Meganathan**, Chairperson **Dr.Thangam Meganathan** and our Principal **Dr.S.N.Murugesan** for providing all the facilities which are required to carry out this project work.

We are ineffably indebted to our H.O.D **Mr.UmaMaheswar Rao.,** for his conscientious guidance and encouragement to make this project a recognizable one.

We are extremely thankful to our faculty **Mr.Vijaykumar M.Tech.,** for his valuable guidance and indefatigable support and extend our heartfelt thanks to all the teaching and non-teaching staff of **Computer Science and Design department** who helped us directly or indirectly in the completion of this project successfully.

At last but not least gratitude goes to our friends who helped us compiling the project and finally to god who made all things possible.

Any omission in this brief acknowledgement doesn't mean lack of gratitude.

AVULA SNEYA DRITI 221701033

LIVIA MARY SEBASTIAN 221701033

MANJUSHREE M 221701036

SNEHA S 221701055

ABSTRACT

The objective of this project is to develop an efficient and user-friendly system for movie ticket booking. The application streamlines the process of selecting movies, showtimes, and available seats, ensuring seamless seat management. It includes features like dynamic seat availability updates, where booked seats are marked as unavailable to prevent double bookings. The system also facilitates secure user registration, login, and transaction management. By integrating real-time booking and administration functionalities, this project aims to enhance user convenience, minimize booking conflicts, and improve operational efficiency in the movie ticketing process.

TABLE OF CONTENTS

CHAPTER – 1

INTRODUCTION

1. INTRODUCTION

The project enables users to efficiently book movie tickets and access essential information about seat availability and bookings. It provides real-time updates, ensuring convenience for users while offering a seamless ticket reservation experience.

2. SCOPE OF THE WORK

The movie ticket booking system simplifies the process of reserving cinema seats, catering to the increasing demand for hassle-free and efficient ticket booking services. It offers quick access and enhanced usability for a diverse range of moviegoers, ensuring a user-friendly experience.

3. PROBLEM STATEMENT

Many users face challenges in reserving movie tickets due to overcrowded booking platforms or unorganized booking systems at cinemas. This results in inconvenience, missed opportunities to secure desired seats, and dissatisfaction among moviegoers. The need for an efficient and accessible system is paramount to addressing these challenges.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The main objective of this project is to allocate available movie seats based on customer requirements while preventing double bookings. The system ensures real-time seat availability updates, tracks bookings effectively, and enhances the overall user experience. Additionally, it aims to streamline operations for cinemas and provide an edge over competitors through efficient service.

CHAPTER – 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	Intel i5
Memory Size	:	8GB (Minimum)
HDD	:	1 TB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 10
Front – End	:	Python (Tkinter for GUI)
Back - End	:	SQLite
Language	:	python,SQL

CHAPTER - 3

MODULE DESCRIPTION

This application consists of two modules. Upon launching, the program will present a login window where the user can log in either as an Administrator or a User. The description of the modules is as follows:

1. Admin Login

When logging in as an Admin, the user must provide valid credentials (username and password). The administrator holds the authority to update, modify, or manage the data stored in the database, ensuring the system runs smoothly and efficiently.

2. User Login

When logging in as a User, the individual can access functionalities such as viewing seat availability, booking tickets, and confirming reservations. User interactions are processed and reflected in the system, ensuring seamless ticket booking.

CHAPTER - 4

SAMPLE CODING

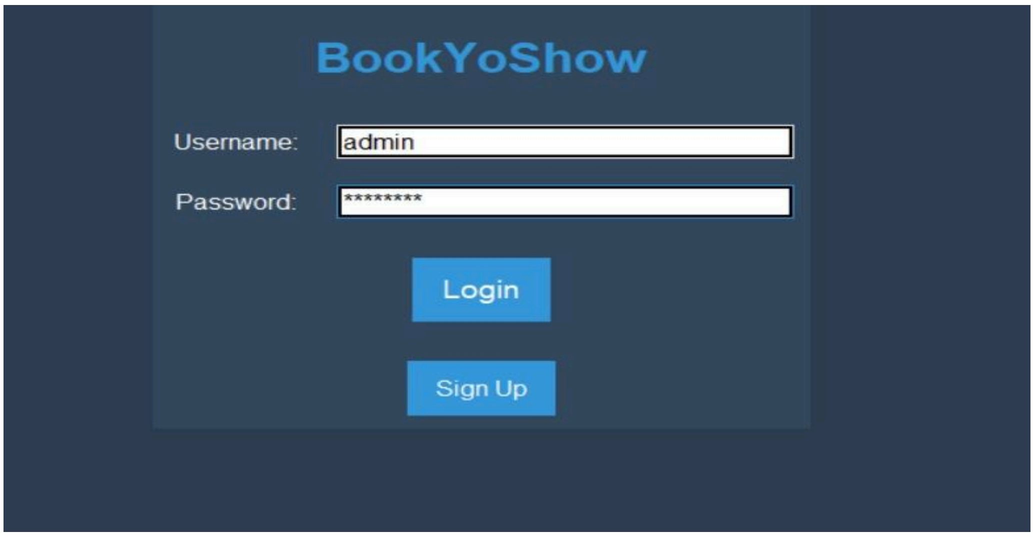
```
import sqlite3
import tkinter as tk
from tkinter import messagebox

class MovieBookingApp:
    def __init__(self, root):
        self.root = root
        self.root.title("BookYoShow - Movie Ticket Booking")


        self.root.geometry("1000x800")  # Full window size
        self.root.configure(bg="#2C3E50")  # Dark background color

        self.login_frame = tk.Frame(self.root, bg="#34495E")
        self.booking_frame = tk.Frame(self.root, bg="#34495E")
        self.theater_frame = tk.Frame(self.root, bg="#34495E")
        self.confirmation_frame = tk.Frame(self.root, bg="#34495E")

        # Application Title
        tk.Label(self.login_frame, text="BookYoShow", fg="#3498DB", bg="#34495E",
font=("Arial", 24, "bold")).grid(row=0, column=0, columnspan=2, pady=20)

        # Create Login Page
        self.create_login_page()

    def create_login_page(self):
        self.login_frame.pack(pady=50, padx=50)

        # Username Label
        tk.Label(self.login_frame, text="Username:", fg="white", bg="#34495E", font=("Arial",
12)).grid(row=1, column=0, padx=10, pady=10)
        self.username_entry = tk.Entry(self.login_frame, width=30, font=("Arial", 12), bd=2,
relief="solid", highlightthickness=1, highlightcolor="#3498DB")
        self.username_entry.grid(row=1, column=1, padx=10, pady=10)

        # Password Label
        tk.Label(self.login_frame, text="Password:", fg="white", bg="#34495E", font=("Arial",
12)).grid(row=2, column=0, padx=10, pady=10)
        self.password_entry = tk.Entry(self.login_frame, show="*", width=30, font=("Arial",
12), bd=2, relief="solid", highlightthickness=1, highlightcolor="#3498DB")
        self.password_entry.grid(row=2, column=1, padx=10, pady=10)

        # Login Button
        login_button = tk.Button(self.login_frame, text="Login", font=("Arial", 14),
bg="#3498DB", fg="white", relief="flat", command=self.check_login)
```

```
        login_button.grid(row=3, column=0, columnspan=2, pady=20, ipadx=10, ipady=5)

        # Sign Up Button
        signup_button = tk.Button(self.login_frame, text="Sign Up", font=("Arial", 12),
bg="#3498DB", fg="white", relief="flat", command=self.show_signup_page)
        signup_button.grid(row=4, column=0, columnspan=2, pady=10, ipadx=10, ipady=5)

    def check_login(self):
        username = self.username_entry.get()
        password = self.password_entry.get()

        if not username or not password:
            messagebox.showerror("Input Error", "Please enter both username and password!")
            return

        # Check if user exists in the database and determine if they're an admin
        conn = sqlite3.connect('movie_booking_system.db')
        cursor = conn.cursor()
        cursor.execute("SELECT is_admin FROM users WHERE username=? AND
password=?", (username, password))
        result = cursor.fetchone()
        conn.close()

        if result:
            self.is_admin = result[0]  # 1 if admin, 0 otherwise
            self.login_frame.pack_forget()
            self.create_theater_selection_page()
        else:
            messagebox.showerror("Login Error", "Invalid username or password!")

    def show_signup_page(self):
        # Create a new window for signup
        signup_window = tk.Toplevel(self.root)
        signup_window.title("Sign Up")
        signup_window.geometry("400x400")
        signup_window.configure(bg="#34495E")

        tk.Label(signup_window, text="Sign Up", fg="#3498DB", bg="#34495E",
font=("Arial", 20, "bold")).pack(pady=20)

        # Username Label and Entry
        tk.Label(signup_window, text="Username:", fg="white", bg="#34495E", font=("Arial",
12)).pack(pady=10)
        username_entry = tk.Entry(signup_window, width=30, font=("Arial", 12))
        username_entry.pack(pady=10)

        # Password Label and Entry
        tk.Label(signup_window, text="Password:", fg="white", bg="#34495E", font=("Arial",
12)).pack(pady=10)
        password_entry = tk.Entry(signup_window, show="*", width=30, font=("Arial", 12))
        password_entry.pack(pady=10)

        # Sign Up Button in the signup window
        def register_user():
            username = username_entry.get()
            password = password_entry.get()

            if not username or not password:
                messagebox.showerror("Input Error", "Please enter both username and password!")
                return

            # Save the user details to the database
            conn = sqlite3.connect('movie_booking_system.db')
            cursor = conn.cursor()
            try:
                cursor.execute("INSERT INTO users (username, password, is_admin) VALUES (?,
?, 0)", (username, password))
                conn.commit()
                messagebox.showinfo("Success", "Registration successful! Please log in.")
                signup_window.destroy()
            except sqlite3.IntegrityError:
                messagebox.showerror("Error", "Username already exists. Please choose a different
username.")
            finally:
                conn.close()

        signup_button = tk.Button(signup_window, text="Register", font=("Arial", 14),
bg="#3498DB", fg="white", relief="flat", command=register_user)
        signup_button.pack(pady=20, ipadx=10, ipady=5)

    def create_theater_selection_page(self):
        self.booking_frame.pack_forget()  # Hide any previous frames
        self.theater_frame.pack(pady=50)

        tk.Label(self.theater_frame, text="BookYoShow", fg="#3498DB", bg="#34495E",
font=("Arial", 24, "bold")).grid(row=0, column=0, columnspan=2, pady=20)

        # Show Add/Delete Theater options if user is admin
        if self.is_admin:
            add_theater_button = tk.Button(self.theater_frame, text="Add Theater",
font=("Arial", 14), bg="#3498DB", fg="white", relief="flat", command=self.add_theater)
            add_theater_button.grid(row=1, column=0, pady=10, ipadx=10, ipady=5)

            delete_theater_button = tk.Button(self.theater_frame, text="Delete Theater",
font=("Arial", 14), bg="#3498DB", fg="white", relief="flat", command=self.delete_theater)
            delete_theater_button.grid(row=1, column=1, pady=10, ipadx=10, ipady=5)

        # Theater Listbox
        tk.Label(self.theater_frame, text="Select a Theater", fg="white", bg="#34495E",
font=("Arial", 16)).grid(row=2, column=0, padx=10, pady=10)
```

```
        self.theater_listbox = tk.Listbox(self.theater_frame, height=5, width=30, font=("Arial",
12), bd=2, relief="solid", selectbackground="#3498DB", selectmode="single")
        self.theater_listbox.grid(row=3, column=0, columnspan=2, padx=10, pady=10)
        self.populate_theaters()

        # Select Theater Button
        select_theater_button = tk.Button(self.theater_frame, text="Select Theater",
font=("Arial", 14), bg="#3498DB", fg="white", relief="flat", command=self.show_movies)
        select_theater_button.grid(row=4, column=0, columnspan=2, pady=20, ipadx=10,
ipady=5)

    def add_theater(self):
        def save_theater():
            name = theater_name_entry.get()
            if not name:
                messagebox.showerror("Input Error", "Please enter a theater name!")
                return

            conn = sqlite3.connect('movie_booking_system.db')
            cursor = conn.cursor()
            cursor.execute("INSERT INTO theaters (name) VALUES (?)", (name,))
            conn.commit()
            conn.close()

            messagebox.showinfo("Success", f"Theater '{name}' added successfully!")
            theater_window.destroy()
            self.populate_theaters()

        # Add Theater Window
        theater_window = tk.Toplevel(self.root)
        theater_window.title("Add Theater")
        theater_window.geometry("300x200")
        theater_window.configure(bg="#34495E")

        tk.Label(theater_window, text="Enter Theater Name:", fg="white", bg="#34495E",
font=("Arial", 12)).pack(pady=20)
        theater_name_entry = tk.Entry(theater_window, font=("Arial", 12), width=20)
        theater_name_entry.pack(pady=10)

        save_button = tk.Button(theater_window, text="Save", font=("Arial", 12),
bg="#3498DB", fg="white", command=save_theater)
        save_button.pack(pady=10)

    def delete_theater(self):
        selected_theater = self.theater_listbox.get(tk.ACTIVE)
        if not selected_theater:
            messagebox.showerror("Selection Error", "Please select a theater to delete!")
            return

        conn = sqlite3.connect('movie_booking_system.db')
        cursor = conn.cursor()
        cursor.execute("DELETE FROM theaters WHERE name=?", (selected_theater,))
        conn.commit()
        conn.close()

        messagebox.showinfo("Success", f"Theater '{selected_theater}' deleted successfully!")
        self.populate_theaters()

    def populate_theaters(self):
        self.theater_listbox.delete(0, tk.END)
        conn = sqlite3.connect('movie_booking_system.db')
        cursor = conn.cursor()
        cursor.execute("SELECT name FROM theaters")
        theaters = cursor.fetchall()
        conn.close()

        for theater in theaters:
            self.theater_listbox.insert(tk.END, theater[0])

    def show_movies(self):
        selected_theater_index = self.theater_listbox.curselection()
        if not selected_theater_index:
            messagebox.showerror("Selection Error", "Please select a theater!")
            return

        selected_theater_id = selected_theater_index[0] + 1

        # Clear previous selections (if any)
        for widget in self.theater_frame.winfo_children():
            widget.grid_forget()

        # Movie Label
        tk.Label(self.theater_frame, text="Select a Movie", fg="white", bg="#34495E",
font=("Arial", 16)).grid(row=0, column=0, padx=10, pady=10)

        # Movie Listbox
        self.movie_listbox = tk.Listbox(self.theater_frame, height=5, width=30, font=("Arial",
12), bd=2, relief="solid", selectbackground="#3498DB", selectmode="single")
        self.movie_listbox.grid(row=1, column=0, padx=10, pady=10)
        self.populate_movies(selected_theater_id)

        # Select Movie Button
        select_movie_button = tk.Button(self.theater_frame, text="Select Movie", font=("Arial",
14), bg="#3498DB", fg="white", relief="flat", command=self.show_showtimes)
        select_movie_button.grid(row=2, column=0, pady=20, ipadx=10, ipady=5)

    def populate_movies(self, theater_id):
        conn = sqlite3.connect('movie_booking_system.db')
        cursor = conn.cursor()
        cursor.execute("SELECT title FROM movies WHERE theater_id=?", (theater_id,))
        movies = cursor.fetchall()
        conn.close()

        for movie in movies:
            self.movie_listbox.insert(tk.END, movie[0])

    def show_showtimes(self):
        selected_movie_index = self.movie_listbox.curselection()
        if not selected_movie_index:
            messagebox.showerror("Selection Error", "Please select a movie!")
            return

        movie_title = self.movie_listbox.get(selected_movie_index)

        # Clear previous selections (if any)
        for widget in self.theater_frame.winfo_children():
            widget.grid_forget()

        # Showtime Label
        tk.Label(self.theater_frame, text="Select Showtime", fg="white", bg="#34495E",
font=("Arial", 16)).grid(row=0, column=0, padx=10, pady=10)

        # Showtime Listbox
        self.showtime_listbox = tk.Listbox(self.theater_frame, height=5, width=30,
font=("Arial", 12), bd=2, relief="solid", selectbackground="#3498DB", selectmode="single")
        self.showtime_listbox.grid(row=1, column=0, padx=10, pady=10)
        self.populate_showtimes(movie_title)

        # Select Showtime Button
        select_showtime_button = tk.Button(self.theater_frame, text="Select Showtime",
font=("Arial", 14), bg="#3498DB", fg="white", relief="flat", command=self.show_seating)
        select_showtime_button.grid(row=2, column=0, pady=20, ipadx=10, ipady=5)

    def populate_showtimes(self, movie_title):
        conn = sqlite3.connect('movie_booking_system.db')
        cursor = conn.cursor()
        cursor.execute("SELECT time FROM showtimes WHERE movie_id=(SELECT
movie_id FROM movies WHERE title=?)", (movie_title,))
        showtimes = cursor.fetchall()
        conn.close()

        for showtime in showtimes:
            self.showtime_listbox.insert(tk.END, showtime[0])

    def show_seating(self):
        selected_showtime_index = self.showtime_listbox.curselection()
        if not selected_showtime_index:
            messagebox.showerror("Selection Error", "Please select a showtime!")
            return
        showtime = self.showtime_listbox.get(selected_showtime_index)

        # Clear previous selections (if any)
        for widget in self.theater_frame.winfo_children():
            widget.grid_forget()

        # Seating Label
        tk.Label(self.theater_frame, text="Select Your Seat", fg="white", bg="#34495E",
font=("Arial", 16)).grid(row=0, column=0, padx=10, pady=10)

        # Seating grid (example for 5x5 seating)
        self.selected_seats = []  # List to store selected seats
        self.seat_buttons = []
        self.load_booked_seats()


    def load_booked_seats(self):
        conn = sqlite3.connect('movie_booking_system.db')
        cursor = conn.cursor()

        # Get booked seats for the selected showtime
        cursor.execute("""SELECT seat_row, seat_column
                          FROM bookings
                          WHERE showtime_id=(SELECT showtime_id
                                             FROM showtimes
                                             WHERE time=?)""",
                       (self.showtime_listbox.get(tk.ACTIVE),))
        booked_seats = cursor.fetchall()
        conn.close()

        # Create seating grid
        for i in range(5):
            row_buttons = []
            for j in range(5):
                seat_button = tk.Button(self.theater_frame, text=f"{i+1}-{j+1}", font=("Arial",
12), bg="#3498DB", fg="white", relief="flat", command=lambda r=i, c=j: self.select_seat(r,
c))

                if (i, j) in booked_seats:
                    seat_button.config(bg="red", state="disabled")
                row_buttons.append(seat_button)
                seat_button.grid(row=i+1, column=j, padx=5, pady=5)
            self.seat_buttons.append(row_buttons)

        book_button = tk.Button(self.theater_frame, text="Proceed to Book", font=("Arial", 14),
bg="#3498DB", fg="white", relief="flat", command=self.book_ticket)
        book_button.grid(row=6, column=0, pady=20, ipadx=10, ipady=5)

    def select_seat(self, row, col):
        conn = sqlite3.connect('movie_booking_system.db')
        cursor = conn.cursor()

        # Check if seat is already booked
        cursor.execute("""SELECT COUNT(*) FROM bookings
                          WHERE showtime_id=(SELECT showtime_id FROM showtimes WHERE
time=?)
                          AND seat_row=? AND seat_column=?""",
                       (self.showtime_listbox.get(tk.ACTIVE), row, col))
        seat_count = cursor.fetchone()[0]
        conn.close()

        if seat_count > 0:
            messagebox.showerror("Booking Error", "This seat has already been booked. Please
select another seat.")
            return

        # Toggle seat selection if it is available
        seat = self.seat_buttons[row][col]
        if seat.cget("bg") == "green":
            seat.config(bg="#3498DB")
            self.selected_seats.remove((row, col))
        else:
            seat.config(bg="green")
            self.selected_seats.append((row, col))

    def book_ticket(self):
        if not self.selected_seats:
            messagebox.showerror("Booking Error", "No seats selected!")
            return

        customer_name = self.username_entry.get()

        conn = sqlite3.connect('movie_booking_system.db')
        cursor = conn.cursor()

        cursor.execute("SELECT showtime_id FROM showtimes WHERE time=?",
(self.showtime_listbox.get(tk.ACTIVE),))
        showtime_id = cursor.fetchone()[0]

        for seat in self.selected_seats:
            seat_row, seat_column = seat

            cursor.execute("""SELECT COUNT(*) FROM bookings
                              WHERE showtime_id=? AND seat_row=? AND seat_column=?""",
                           (showtime_id, seat_row, seat_column))
            seat_count = cursor.fetchone()[0]
            if seat_count > 0:
                messagebox.showerror("Booking Error", f"Seat {seat_row+1}-{seat_column+1}
has just been booked. Please reselect seats.")
                self.selected_seats.clear()
                self.show_seating()
                conn.close()
                return

            cursor.execute("""INSERT INTO bookings (customer_name, showtime_id, seat_row,
seat_column)
                              VALUES (?, ?, ?, ?)""",
                           (customer_name, showtime_id, seat_row, seat_column))

        conn.commit()
        conn.close()

        messagebox.showinfo("Booking Successful", f"Booking Successful for
{customer_name}.\nSeats: {', '.join([f'{r+1}-{c+1}' for r, c in self.selected_seats])}")
        self.selected_seats.clear()
        self.show_confirmation()

        # Show a success message
        messagebox.showinfo("Booking Successful", f"Booking Successful for
{customer_name}.\nSeats: {', '.join([f'{r+1}-{c+1}' for r, c in self.selected_seats])}")
        self.selected_seats.clear()  # Clear selected seats after booking

        # Show Thank You page
        self.show_confirmation()

    def show_confirmation(self):
        # Hide the current frame
        for widget in self.theater_frame.winfo_children():
            widget.grid_forget()

        # Confirmation message
        tk.Label(self.theater_frame, text="Thank You for Booking!", fg="white",
bg="#34495E", font=("Arial", 24, "bold")).grid(row=0, column=0, pady=20)

        # Return to home page button
        return_button = tk.Button(self.theater_frame, text="Return to Home", font=("Arial",
14), bg="#3498DB", fg="white", relief="flat", command=self.go_home)
        return_button.grid(row=1, column=0, pady=20)

    def go_home(self):
        self.theater_frame.pack_forget()  # Hide current frame
        self.login_frame.pack(pady=50, padx=50)  # Show login frame again

# Create Tkinter window
root = tk.Tk()
app = MovieBookingApp(root)
root.mainloop()
```

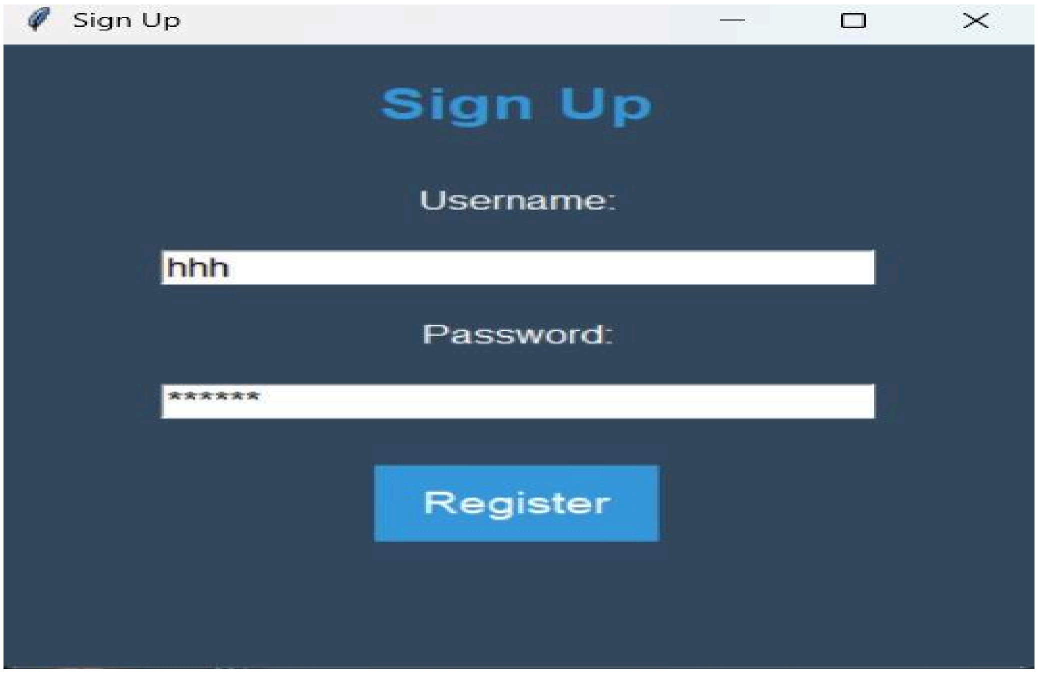
CHAPTER - 5

SCREEN SHOTS



Username: admin
Password: ********

Fig 5.1 Login page



Username: hhh
Password: ******

Fig 5.2 SignUp Page

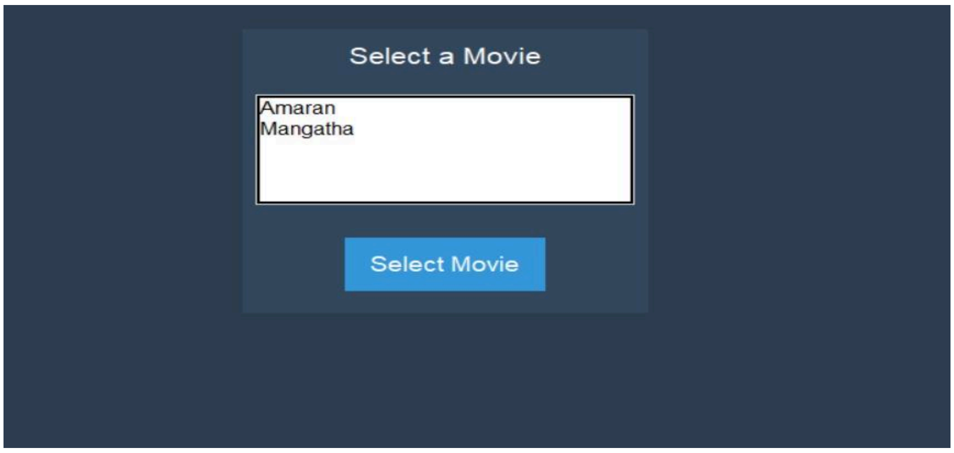


Fig 5.3 Selection of Movie Page

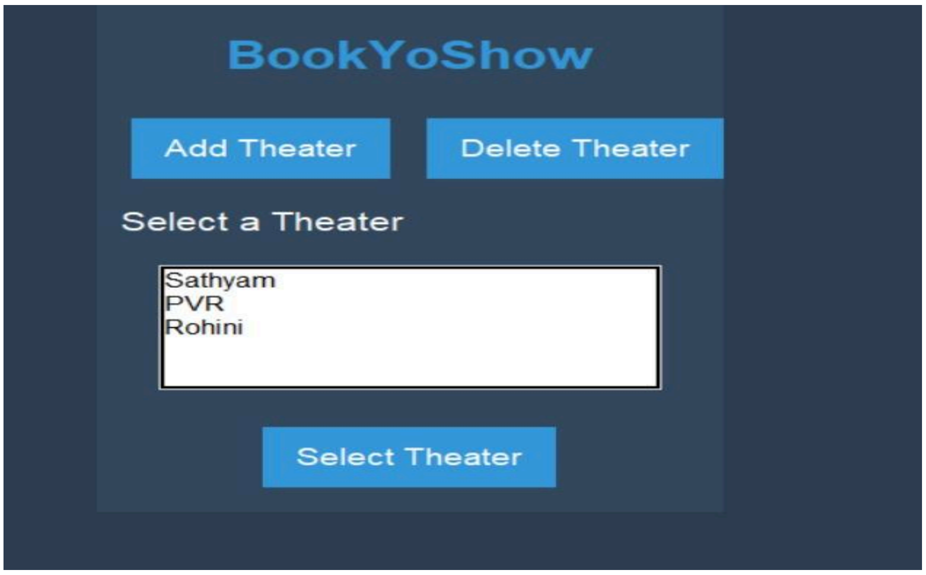


Fig 5.4 Selection of Theatre Page

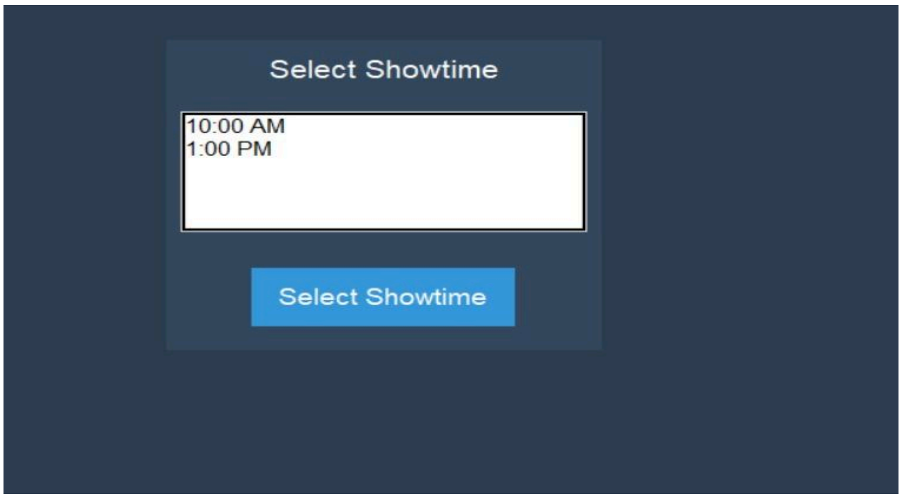


Fig 5.5 Selection of Show Time Page

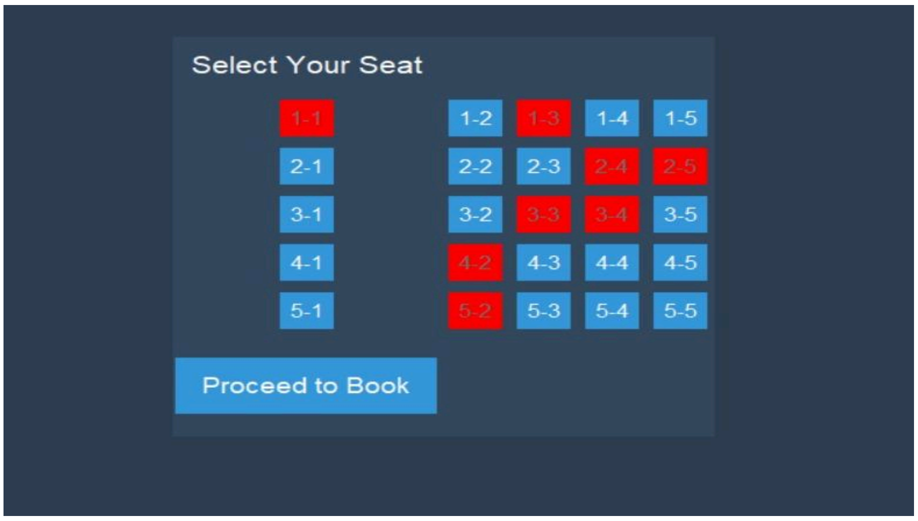
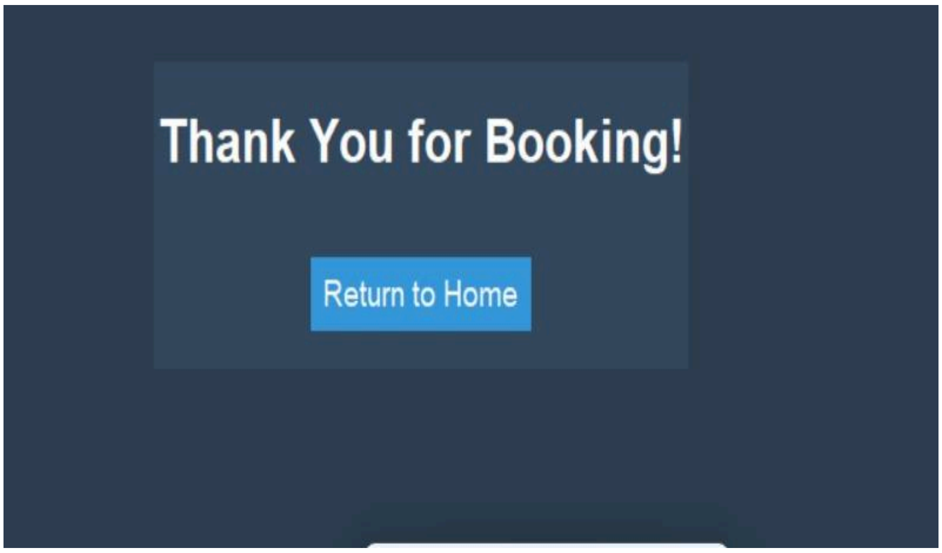
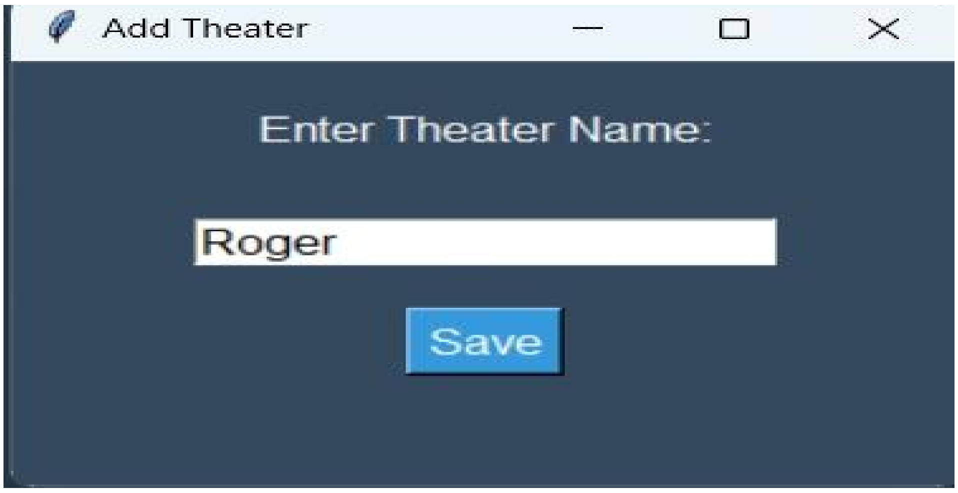


Fig 5.6 Selection of Seat Page



Thank You for Booking!

Fig 5.7 Thank You Page



Enter Theater Name:

Roger

Fig 5.8 Add theatre Page

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

With the implementation of this project, users can easily check seat availability and book movie tickets efficiently. The system simplifies ticket management by maintaining a clear log of bookings and ensuring transparency for both users and administrators. By providing a user-friendly interface and a reliable booking system, this project enhances convenience and accessibility for all users.

In the future, the system can be further enhanced by integrating advanced features like dynamic pricing, multi-language support, and AI-driven recommendations for personalized movie suggestions. These enhancements will broaden accessibility and improve user experience, making the system more versatile and efficient.

CHAPTER – 7

REFERENCES

1. https://www.w3schools.com/sql/
2. https://www.tutorialspoint.com/sqlite/index.htm
3. https://www.wikipedia.org/
4. https://www.learnpython.org/
5. https://www.codecademy.com/learn/learn-python