# Movie Theatre Management System

**A Report for the Evaluation Final Project**

***Submitted by***

## Student Name (Reg#)

***in partial fulfilment for the award of the degree of***

**BACHELOR OF COMPUTER APPLICATION**

**IN**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**Under the Supervision of**

**(Supervisor Name)**

## Assistant/Senior Professor

**MAY- 2023**

|  |
| --- |
| **SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**  **BONAFIDE CERTIFICATE**    Certified that this project report “**Movie Theatre Management System**” is the bonafide work of “**Student Name (Reg#**)” who carried out this project work under my supervision.            **SIGNATURE OF HEAD SIGNATURE OF SUPERVISOR**  **Dr. ABC (Supervisor Name)**  **PhD (Management), PhD(CS) Assistant Professor**  **Professor& Dean, School of Computer Science &**  **School of Computer Science & Engineering**  **Engineering** ACKNOWLEDGEMENT I am extremely grateful and remain indebted to my guide (**Supervisor Name**) for being a source of inspiration and for her constant support in the Design, Implementation and Evaluation of the project. I am thankful to her for her constant constructive criticism and invaluable suggestions, which benefited me a lot while developing the project on “**Movie Ticketing Management System**”. He/She has been constant source of inspiration and motivation for hard work. She has been very co-operative throughout this project work. Through this column, it would be my utmost pleasure to express my warm thanks to her for her encouragement, co- operation and consent without which I might not be able to accomplish this project.  I also express my gratitude to (**Supervisor Name**) for providing me the infrastructure to carry out the project and to all staff members of my collage who were directly and indirectly medium in enabling me to stay committed for the project. |

Problem Analysis:

The Movie Ticketing Management System is designed to automate the process of buying movie tickets. The system should allow customers to browse available movies, select seats, make payments, and receive tickets electronically. The system should also allow employees to manage movie showtimes, update ticket prices, and view customer information.

**Modules:**

The following modules are used in the Movie Ticketing Management System:

**Module Name:** MainPage

**Class Name:** MainPage

**Purpose:** This Class defines a GUI application's main page, which allows the user to select their role from three options: Manager, Staff, or Customer. Depending on the selection, the application opens a new window (Manager, Staff, or Customer) with its own layout and functionality, and the main page window is hidden. The buttons change color when the mouse is over them. There is also a Quit button to close the application.

**Module Name:** Customer

**Class Name:** Customer

**Purpose:** This class named "Customer" which is a login page for customers. It creates a GUI with a username and password entry fields, a "Keep me logged in" checkbox, and three buttons for logging in, going back, and creating a new account. When the user clicks the login button, it checks if the entered username and password match with the data stored in the database. If the login credentials are correct, it opens a new window and shows the main page for the customer. If the login credentials are incorrect, it shows an error message. The create button opens a new window for creating a new account, and the back button takes the user back to the main page.

**Module Name:** Customer Main Page

**Class Name:** CustMain

**Purpose:** The code defines a class called CustMain which inherits from another class called Customer. It creates a customer main menu page that allows the user to update their profile, make a ticket booking, view booking details, and check ticket availability and seat selection. The sch method retrieves data from a database based on a selected date and displays the results in a new window. The other methods Update\_Profile\_window, Booking\_History\_window, and logout open different windows based on user actions.

**Module Name:** Search Results

**Class Name:** SearchResults

**Purpose:** The class named SearchResults which is a subclass of CustMain. It has an initializer which initializes some instance variables.

The class also defines a method boo which is executed when the user wants to book a ticket for a movie. The method takes some inputs from the user and updates the booking details in the database.

There is also a back method which takes the user back to the previous screen. The class also creates a GUI where users can search for movies based on the date and time, and book tickets for the available seats.

**Module Name:** Update Customer Profile

**Class Name:** CustProfile

**Purpose:** This Python class creates a window for customers to update their profile information. It includes fields for the customer's first name, last name, email address, age, and password. The customer can edit these fields and click the "Update Details" button to save the changes. The class validates the inputs and displays an error message if the inputs are not valid. If the changes are saved successfully, a success message is displayed, and the window is closed. The class also includes a "Back" button that returns the user to the previous window and a "Logout" button that logs the user out. The code uses the tkinter module for creating GUIs.

**Module Name:** Create New Account

**Class Name:** NewAccount

**Purpose:** This is a NewAccount class that inherits from a Customer class. It creates a GUI that allows users to enter their details to create a new account. The GUI includes fields for the user's first name, last name, email address, age, username, and password. Once the user has filled out all of the fields and clicks the "Create new account" button, the code validates the input and inserts the user's information into a database table named "customers" if the input is valid. If the input is invalid, the code displays an error message indicating what went wrong. The code also includes a "Back" button that allows the user to return to the previous screen.

**Module Name:** Book Seats

**Class Name:** Booked

**Purpose:** This is a class Booked that inherits from CustProfile and SearchResults. It initializes several labels and buttons for a booking confirmation page, retrieves customer details from a database, and inserts a new booking into a separate database. The booking details include the customer's name, email, age, the date and time of the booking, and the assigned seat number.

**Module Name:** Booking History

**Class Name:** BookHist

**Purpose:** This class BookHist inherits from a class called Booked. It creates a window that displays the booking history of the logged-in user.

The \_\_init\_\_ method initializes various labels and buttons, as well as queries the database for the user's booking history. It then populates the window with the booking history data. Each row displays the date, time, movie title, seat number, and a remove button.

The remove method is used to remove a booking from the user's booking history. It asks for confirmation, checks if the date and time of the booking has already passed, and then removes the booking from the database. It then updates the movie database to reflect the change in the number of available seats. Finally, it displays a message box to confirm that the booking has been removed.

**Module Name:** Manager Login

**Class Name:** Manager

**Purpose:** This class Manager inherits from the Customer class. The Manager class has an \_\_init\_\_ method that calls the \_\_init\_\_ method of the parent class. The Manager class also has a login method that reads a file named manager.txt and compares the username and password entered by the user to the data in the file. If the username and password match, it opens a new window with the ManagerMain class and displays a success message.

**Module Name:** Manager Main Page

**Class Name:** ManagerMain

**Purpose:** This class ManagerMain represents a graphical user interface (GUI) for a movie management system. The GUI has four buttons: "Add Movie", "Schedule Shows", "Add a Customer as a Member", and "Logout". The add\_movie\_func function creates a new window for adding movie details and submits them to a database using the add\_movie\_to\_database function. The logout function prompts the user to confirm that they want to log out before navigating to the main page. The Add\_film\_showings function navigates to a window for scheduling showings of movies. The Add\_member function creates a new window for adding a customer as a member to the system.

**Module Name:** View Booking Details

**Class Name:** Showings

**Purpose:** This is a class called Showings, which is a child class of ManagerMain. It displays a list of movies with their date, time, title, description, booked seats, available seats, and a button to see the seating arrangement. The movies are ordered by date and time, with the earliest dates and times first. There are buttons to export the movies to a text file, go back to the previous screen, and log out of the system. When the export button is clicked, a text file called "export.txt" is created, and a list of movies with their date, time, title, booked seats, and available seats is written to the file.

**Module Name:** Add Movie

**Class Name:** AddShowings

**Purpose:** This is a Python class AddShowings, which inherits from the Showings class. The AddShowings class has a constructor that creates a frame, label, and various form elements such as labels, option menus, text boxes, and buttons for adding a new showing. When the user clicks the "Add Film Showing" button, the code reads the values of the form elements, validates them, and inserts a new showing into the list of showings if the values are valid. If the values are not valid, an error message is displayed.

**Module Name:** View Seats Empty

**Class Name:** SeeSeatsEmp

**Purpose:** This class SeeSeatsEmp inherits from another class Showings. The SeeSeatsEmp class displays a seat layout for a movie theater, with green buttons representing available seats and red buttons representing taken seats. The number of taken seats is provided in the booked parameter. The layout is created using the Button and Frame widgets from the tkinter library. The layout is divided into 10 rows of seats, each with 10 seats, except for the last row, which may have fewer seats. The back and log buttons are also created to allow the user to return to the previous screen or to log out.

**Module Name:** Staff Login

**Class Name:** Staff

**Purpose:** This is a class representing the login page for staff members. It inherits from the Customer class, which likely includes some shared functionality with regular customers. The login method is called when the staff member attempts to log in with their username and password. The method reads usernames and passwords from a file called staff.txt, checks if the entered credentials match any of the entries in the file, and if a match is found, opens a new window with the StaffMain class (not shown). If there is no match, a warning message is displayed.

**Module Name:** Staff Main Page

**Class Name:** StaffMain

**Purpose:** This StaffMain class inherits from the Staff class and adds the necessary buttons and functions for the staff main menu page. The logout function creates a pop-up message box to confirm if the user wants to log out and then opens the main page (MainPage class) if the user confirms. The See\_list and Add\_film\_showings functions open the respective pages (Showings class and AddShowings class) when their buttons are clicked. Overall, the code follows good object-oriented principles by separating the different functionalities into their respective classes.

**Databases:**

The Customer\_Database\_Initialiser() function initializes the database for customers, creating a table named customers with columns for first name, last name, email, age, username, and password. It also reads from a file named customers.txt and inserts the data into the customers table.

The Films\_Database\_Initialiser() function initializes the database for film showings, creating a table named movies with columns for date, time, title, description, booked, and available. It also reads from a file named MOVIES.txt and inserts the data into the movies table.

The Bookings\_Database\_Initialiser() function initializes the database for film bookings, creating a table named bookings with columns for first name, last name, date, time, seat number, and username.

The main() function initializes the Tkinter window, sets the window title and geometry, and creates an instance of the MainPage class. It also calls the three database initialization functions.

Overall, the code sets up the initial databases and GUI for the movie theatre management system, allowing users to log in, view movie showings, and make bookings.

**Design:**

The UML class diagram for the Movie Ticketing Management System is as follows:

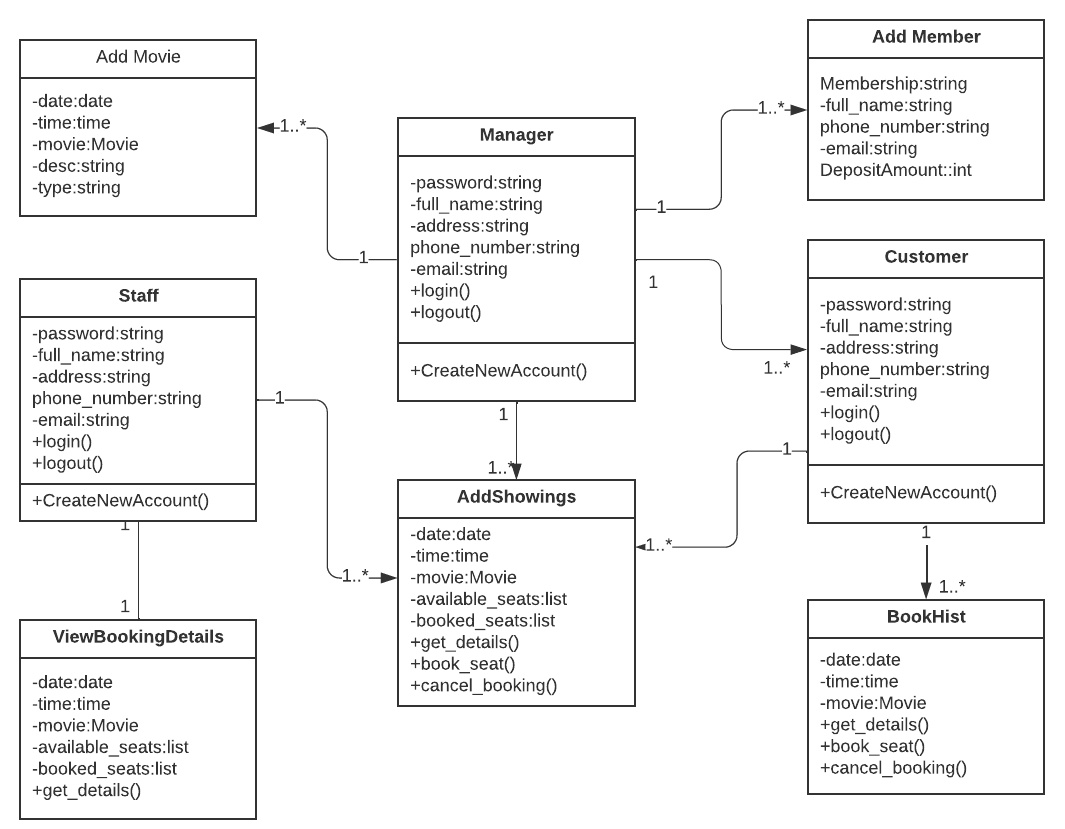


Fig.1: UML class diagram for the Movie Ticketing Management System.

The class diagram shows the relationship between the classes and their attributes/data, behavior/function, class relationships, and cardinality.

Here is the pseudocode or algorithm for the "Movie Theatre Management System":

**Main Page**

* Create the GUI elements of the main page
* Create functions to handle user actions
* Define the logical flow for each action
* Implement error handling for invalid user inputs
* Save any changes to the databases and update the interface as necessary

**Manager Interface:**

**Note:** Manager Login Credentials e.g (Username & Password) are already created & stored in a text file named as “Manager”. Copy credentials and paste them in Login fields to login.

1. Start the program
2. Display main menu with options to add a movie, schedule shows, and add a customer as a member
3. Let the user choose an option
4. If the user chooses to add a movie:
5. Prompt the user to enter the movie title, description, genre, and duration
6. Create a new movie object with the user's input
7. Save the movie object to the database
8. Display a success message
9. If the user chooses to schedule shows:
10. Display a list of movies available in the database
11. Prompt the user to choose a movie to schedule shows for
12. Prompt the user to enter the date and time for the show
13. Create a new show object with the movie, date, and time
14. Save the show object to the database
15. Display a success message
16. If the user chooses to add a customer as a member:
17. Prompt the user to enter the customer's name, email, and phone number
18. Create a new customer object with the user's input
19. Save the customer object to the database
20. Display a success message
21. Display the main menu again
22. If the user chooses to exit the program, terminate the program
23. Otherwise, repeat from step 3

**Staff Interface**

1. Start
2. Initialize the staff class and its attributes
3. Display options for staff to select from:
4. Make a ticket booking for a customer
5. View Booking Details
6. Check ticket availability
7. Seat Selection
8. Get input from staff on which option they want to choose
9. If the staff chooses option a, then
10. Ask for customer details (name, contact, etc.)
11. for movie availability and seat availability
12. If available, book the ticket and provide ticket details to the customer
13. If not available, inform the customer and suggest alternate movie timings or seats
14. If the staff chooses option b, then
15. Ask for booking ID or customer details
16. Display booking details for the requested booking
17. If the staff chooses option c, then
18. Ask for movie details and show timings
19. Check for ticket availability and display available seats
20. If the staff chooses option d, then
21. Ask for movie details and show timings
22. Display the seat map for the selected show
23. Let the staff select the desired seats for the customer
24. End.

**Customers Interface**

* Create a login page for customers
* Verify customers credentials
* Provide options to add, edit or delete film showings and view booking details
* Save any changes to the databases and update the interface as necessary

**Create the main window for the system**

* Initialize the window object
* Set the window title and geometry
* Load a background image
* Create the main page object and place it in the window
* Run the main loop of the window

**Registration Module**

* Provide a registration page for new customers
* Verify that all required fields are filled out correctly
* Generate a unique username and password for the new customer
* Insert the customer data into the customer database
* Update the interface to show the new customer details

**Booking Module**

* Provide a film selection page for customers
* Display available showings and allow customers to select a date and time
* Show available seats and allow customers to select seats
* Allow customers to order refreshments and pay for the booking
* Insert the booking data into the bookings database
* Update the interface to show the new booking details

**Cancellation Module**

* Provide a cancellation page for customers
* Allow customers to select a booking to cancel
* Remove the booking data from the bookings database
* Update the interface to show the cancelled booking details

**Reporting Module**

* Provide a reporting page for employees
* Display various statistics about film showings and bookings
* Allow employees to filter and sort the data as needed

**Initialize the customer database**

* Open a connection to the database
* Create a cursor object
* Create the customers table if it does not exist
* Parse the customer data from a text file and insert it into the table
* Commit changes and close the connection

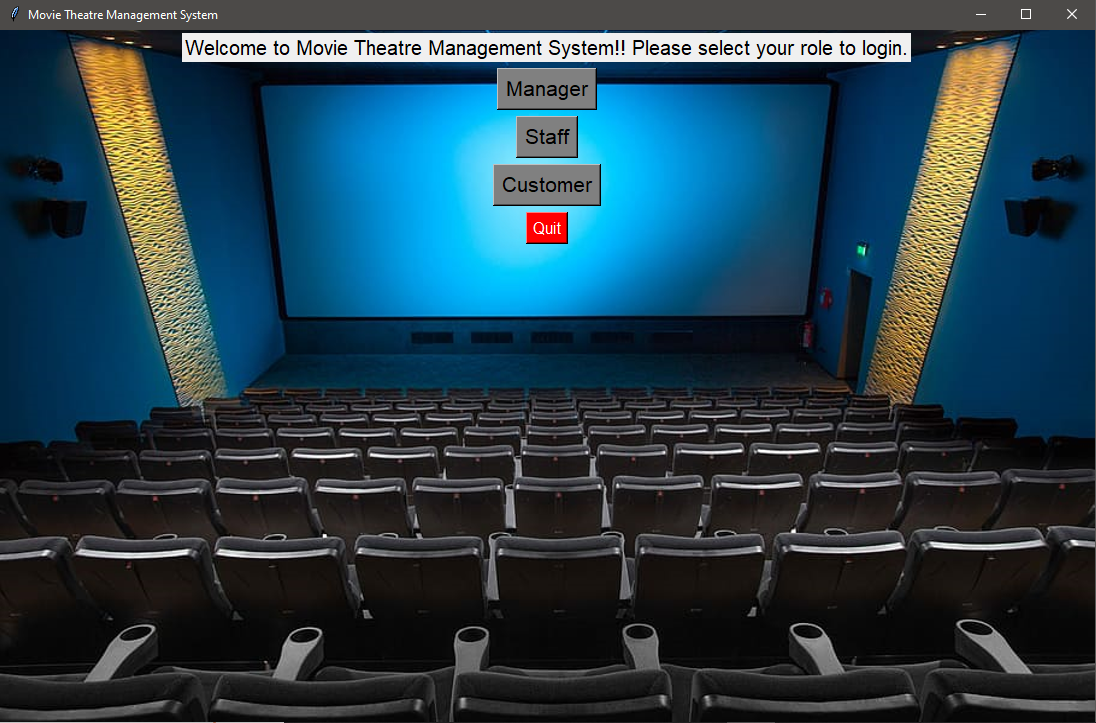
**Initialize the film database**

* Open a connection to the database
* Create a cursor object
* Create the movies table if it does not exist
* Parse the film data from a text file and insert it into the table
* Commit changes and close the connection

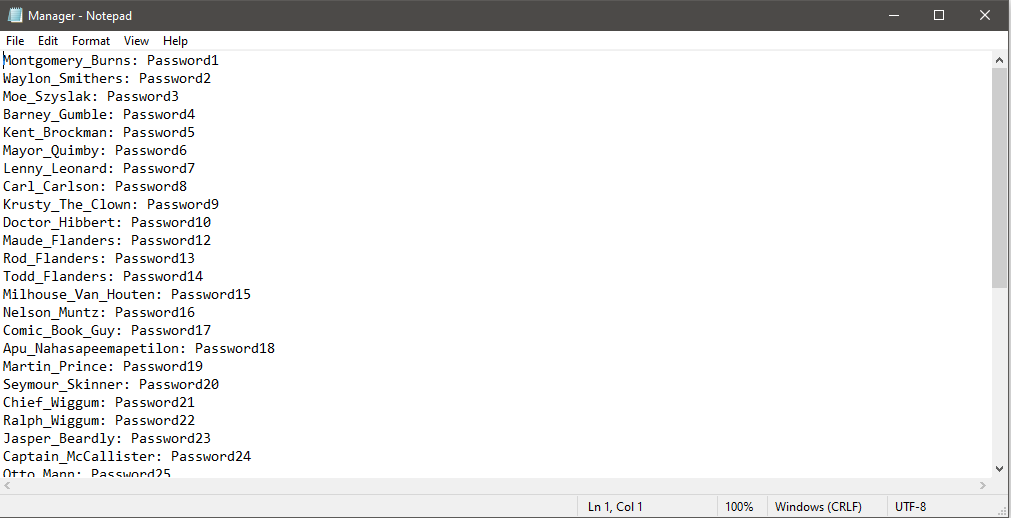
**Initialize the bookings database**

* Open a connection to the database
* Create a cursor object
* Create the bookings table if it does not exist
* Commit changes and close the connection

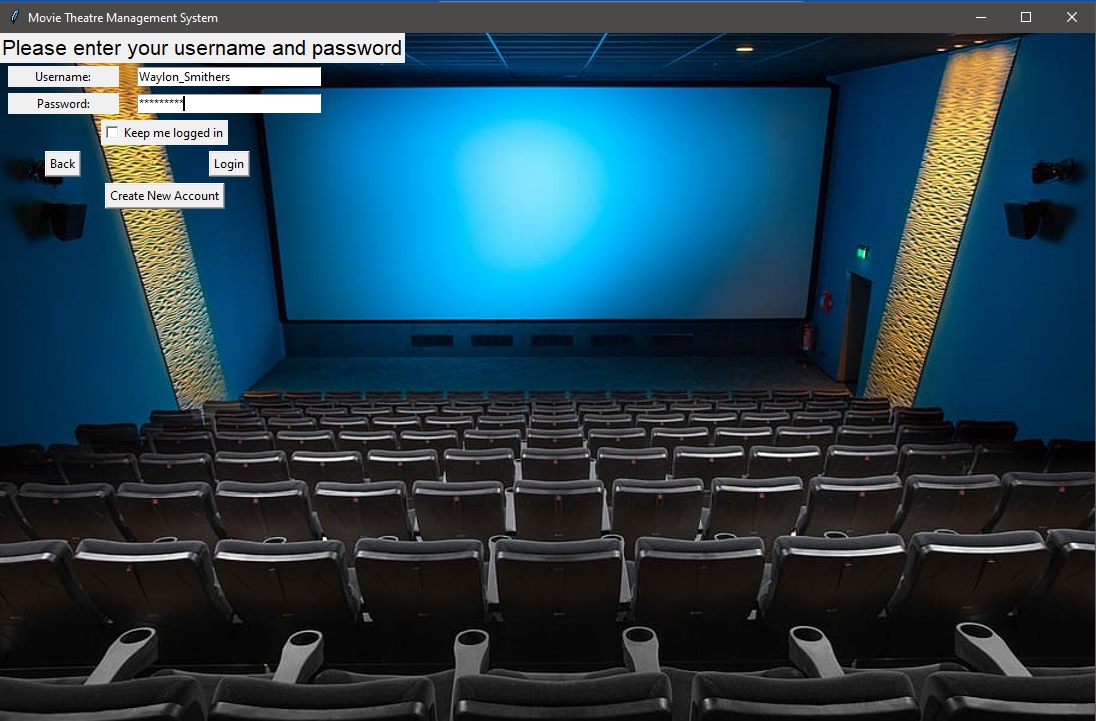
**Main Page:**



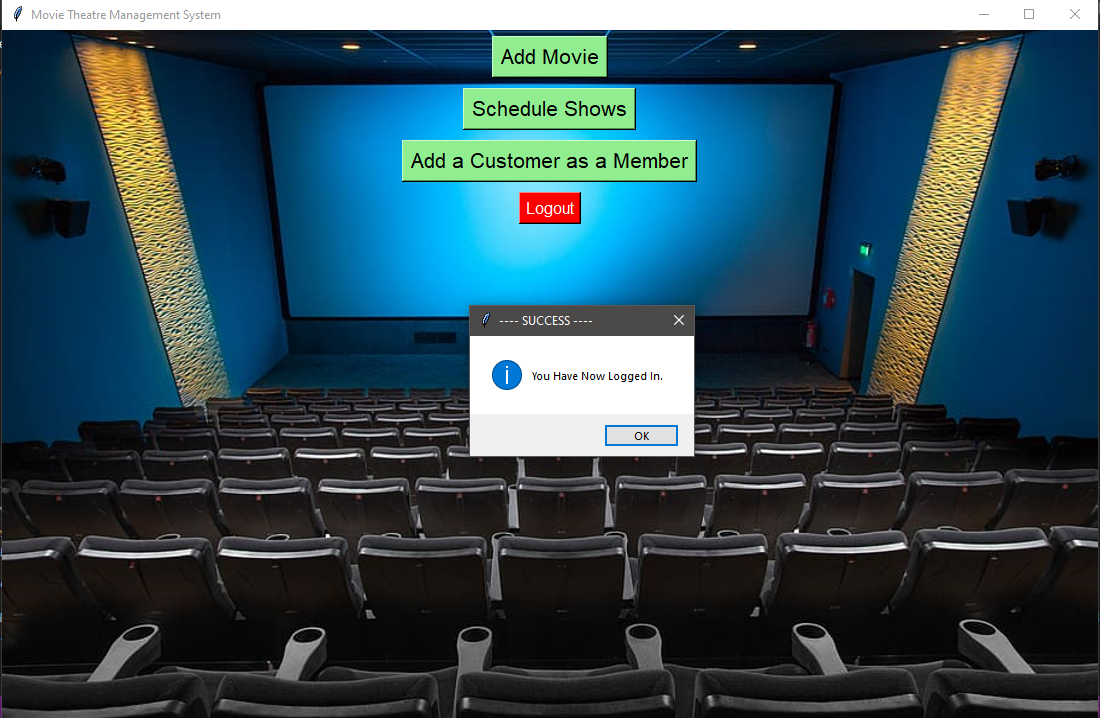
**Manager Credentials:**



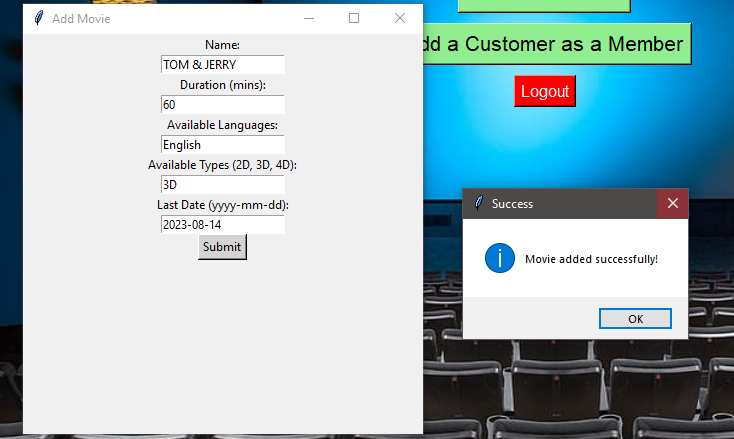
**Login Screen (Manager)**



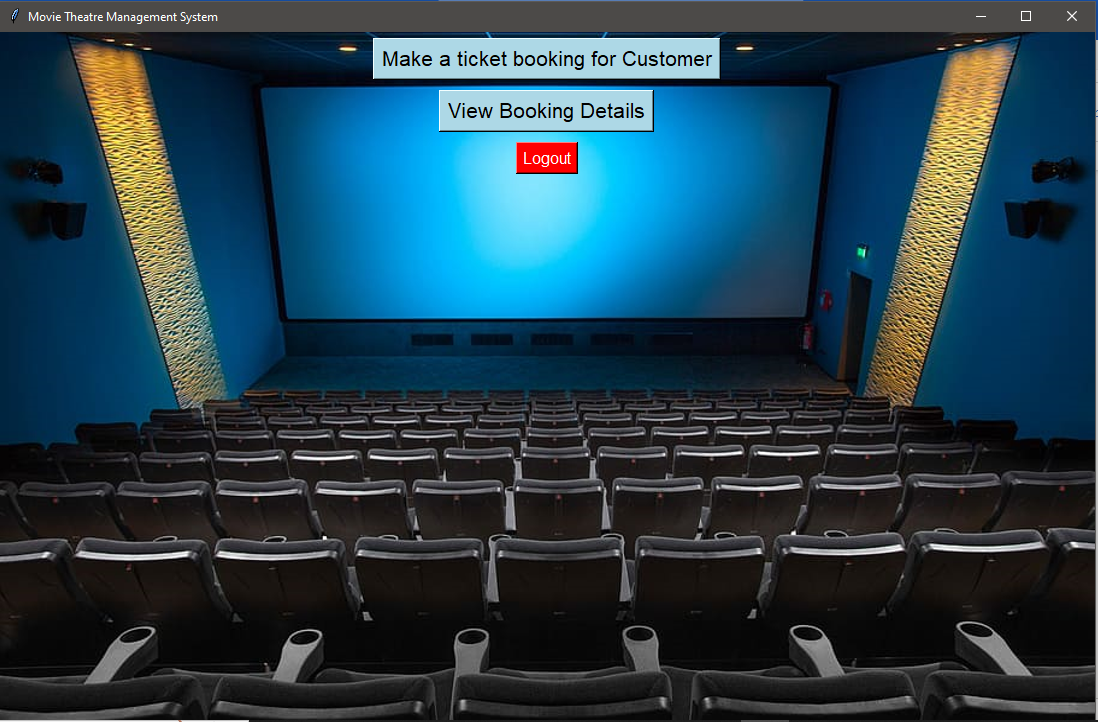
**Manager Profile:**



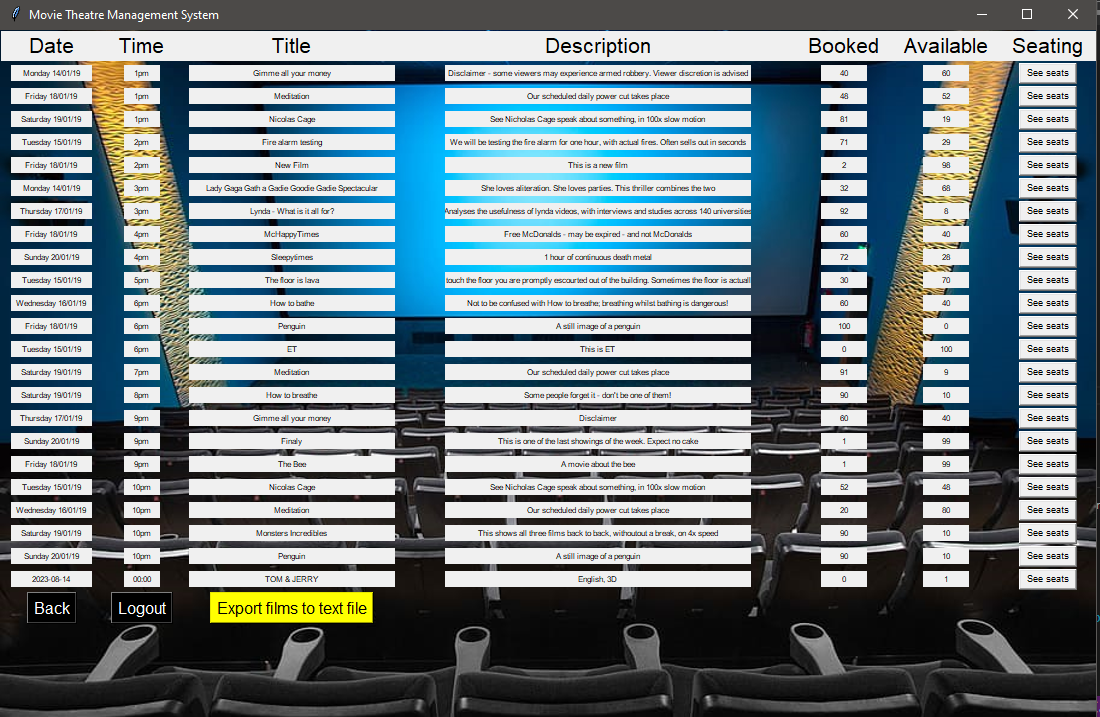
**Add Movies (Manager Profile)**



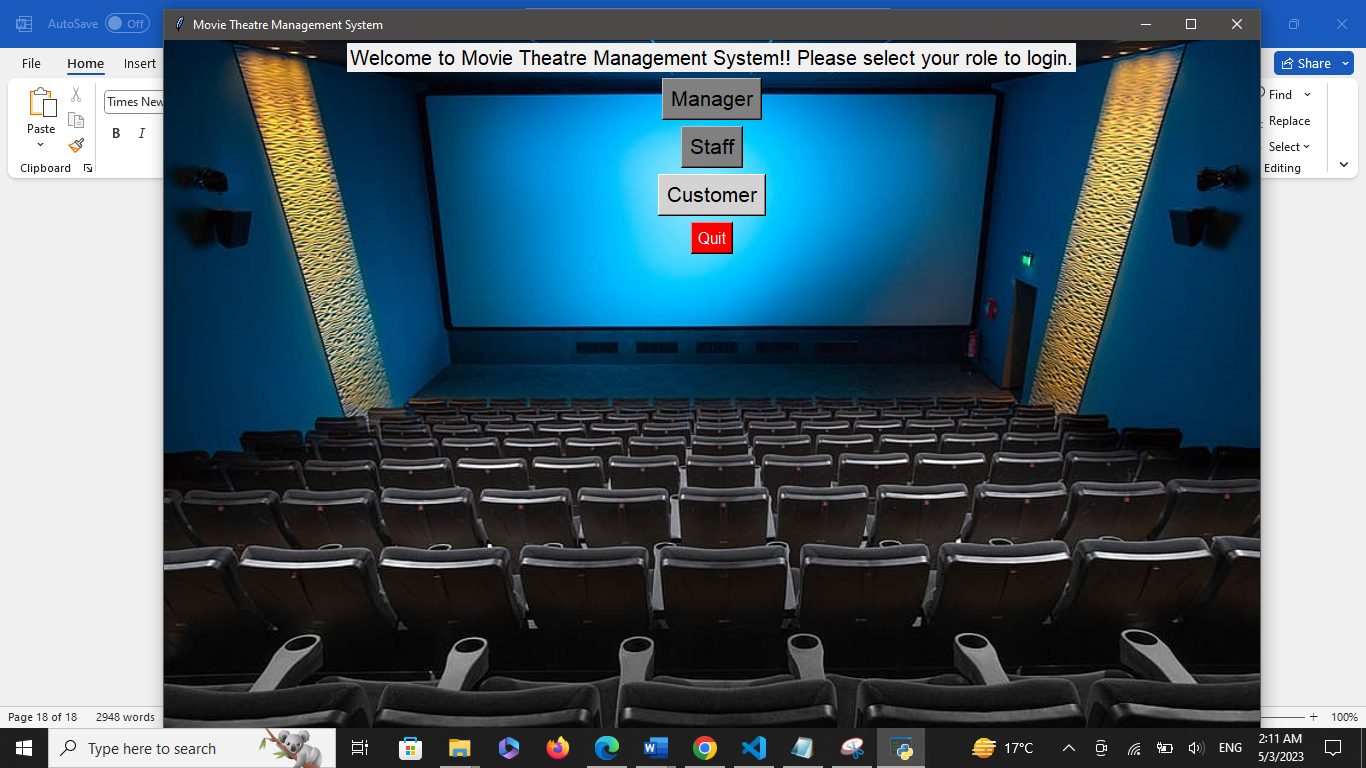
**Staff Profile:**



**View Booking Details (Staff)**



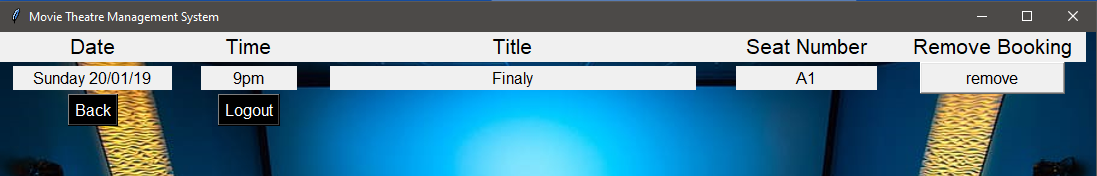
**Customer Window:**



**Customer Login Screen:**



**View Booking Details:**



**Update Profile:**



**Check Availability & Make Booking:**

