CPSC471: Project Final Report

Project Title: Movie Master
Group Number: 41
Members: Jared Assen (30090055)
Madeline Mazurek (30090526)
Date Submitted: April 13th, 2022

1. Abstract

Movie Master is a web-based application that allows customers to make accounts and purchase tickets for movie showings across different theaters and branches. It aims to solve the problem of having a variety of different theater websites to check to find the right movie by consolidating all the theater information in one place. Users are able to filter showings by city name, and theater and branch name. Users can sign up to create an account and then log in to purchase tickets. All users can view movies and add tickets to their cart, but only logged users with stored payment information can proceed to checkout. System administrators have access to the underlying database the application is built on, and can add and/or delete any values stored in the database. This allows admins to populate the database with new movie information and ticket options for customers to purchase. Once a ticket is purchased using stored customer payment information, the ticket is removed from selection to prevent multiple users from buying the same ticket. The front end of the application was built using an angular framework and the backend was designed using an ASP.NET core REST API. Stored procedures were used anytime a customer directly queried the database with free input to prevent SQL injection attacks. Other stored procedures for specialized queries such as searching by a city name were also created to allow for full functionality of the web application.

2. Introduction

2.1. Problem Definition

Before the advent of the technological revolution, movie theaters would operate under a simple system of selling tickets at the door. This system's faults could immediately be seen on a busy day when one would make the trek to the theater only to find out that there were no remaining available seats at that time. At this point, the only options were to leave, see another undesired movie, or come back at a later time and hope there were seats. By moving ticket purchasing online, theaters were able to solve this initial problem by providing a reservation system to customers. Customers could now actively monitor seat numbers, movie showtimes, movie prices, and seat locations. By allowing customers to reserve specific seats online before the movie, theaters became more user-friendly. The overall movie experience improved as customers could now select the exact seats they wanted, and if those seats were taken they could find another time when they would be available. However, there still remains the problem of having to search multiple websites to purchase the best ticket. Presently, there is no mainstream application that provides the exact service this project will solve.

Currently, data on all showtimes in an area can be acquired with a crafty browser search but there is no option to purchase a ticket without first being directed to the specific theater's website. Additionally, this method produces results that are generally disorganized, unintuitive, and not user-friendly. This opens the door for an improvement to the current system as there is still no one unified application for online ticket purchasing. The interest in this project stems from the fact that it is an innovative and creative improvement to the current system. Each and every time that a customer goes to purchase a movie ticket they are met with inconveniences that our developed program solves.

2.2. Developed System

By using a REST API to allow frontend communication with the movie master database, users are provided with all the tools they need to purchase tickets for movies across all theaters in the database. When first entering the website, users are met with a list of all current showings in the database along with a dropdown display that contains additional information about each movie. From here, users are able to search by city and by theater and branch name, to consolidate showings they are interested in. Users can then view and add as many tickets to their cart for as many movie showings as they want. Before checking out the user are prompted to log in which will require them to create an account if one does not yet exist. The account creation prompts users to enter their payment information, allowing for a streamlined and accurate checkout process. By checking out, all tickets in the user's carts are purchased and are removed from the ticket selection page from all other users to prevent duplicate ticket purchases.

Movie Master’s database is designed to hold all showings across all branches for all theaters stored. Each showing also includes all tickets and uses a stored procedure to track ticket purchases. When a customer checks out, the buyerEmail attribute in the ticket tuple is populated with the logged-in user's email, which is the customer entity’s primary key. Next time the user goes to view the same ticket select page for the same showing, an HTTP GET request will be sent using a procedure that will query the database only for tickets with an empty buyerEmail attribute.

By providing users with a ticket purchasing application for all theaters, the problem of having to check multiple specific theater websites for movie showings is solved. Now, instead of having to check every theater website to find the best showing in an area, customers can view showings in a specified city and purchase tickets for any one of them. Movie Master also allows for searching by theater and branch name to allow customers with the same functionality that is provided by current theater websites.

The website also provides system administrators with an interface to add and delete all types of movies, theaters, and showing information. In this way, admins can log in to the application using a specific admin email and password, and populate the database with everything and anything that customers will need to make an informed purchase of tickets.

3. Project Design

3.1. General System User: Provided Functionalities

- Home Page: A customer can view all movie showings in the database by default. They can also select drop-down menus to view more specific movie information such as writer and director names. A customer can also search by city name, and by theater and branch name.

- Ticket Select: A customer can view all available tickets for a particular showing and press a button to add them to their cart. All relevant ticket information will be displayed for customers to see.

- Checkout: Here users will be able to view the tickets in their cart and the total price of all the tickets. They will also be shown their payment information. Users can press a button to clear their cart or to confirm their purchase.

- Purchase Success: Customers will be shown a confirmation page that will show them their email and purchase confirmation.

- Login: Customers with an account will be able to enter their registered email and password to sign in. This will give them access to the checkout page.

- Signout: Customers will be able to press a sign-out button to log out of their account.

- About: Customers will be able to see a short write-up about the website and what it does.

- Signup: Customers can make an account by entering personal information including payment information.

3.2. System Administrator: Provided Functionalities

- Admin login: Admins will be able to enter their admin email and password to sign in to their account. This will give them access to the admin page.

- Signout: Admins can press a sign-out button to log out of their account.

- Admin Page: Admins will have access to add or delete an entry into the database. They will have access to every table and will be able to add and delete tuples from the database.

- Add Theater: Admins can input a theater name to add a theater into the database.

- Add Shows: Admins can select theaters and movies from dropdowns to set a shows entry in the database.

- Add Branch: Admins can add a branch by entering all branch attributes.

- Add Showroom: Admins can add a showing by entering all showing attributes.

- Add Showing: Admins can add a showing by entering all showing attributes.

- Add Ticket: Admins can add tickets by entering all ticket attributes. They will also be able to enter how many tickets they want to add and the ticketID primary keys will be auto-generated for all the tickets.

- Add System Admin: Admins can add an admin account by entering all system admin attributes.

- Add Movie: Admins can add a movie by entering all movie attributes.

- Add Movie Writer: Admins can add a movie writer to a movie by entering all Movie_Writer attributes.

- Add Movie Producer: Admins can add a movie producer to a movie by entering all Movie_Producer attributes.

- Add Movie Cast Member: Admins can add a movie cast member to a movie by entering all Movie_Cast attributes.

- Add Movie Genre: Admins can add a movie genre to a movie by entering all Movie_Genre attributes.

- Add Movie Director: Admins can add a movie director to a movie by entering all Movie_Director attributes.

Note: All foreign key elements when adding are provided to admins to select via drop-down menus to make it more user-friendly.

- Delete Theater: An admin can delete a theater by selecting the theater name from a drop-down menu.

- Delete Shows: An admin can delete a shows tuple by selecting the theater name and movie title from drop-down menus.

- Delete Branch: An admin can delete a branch tuple by selecting the BranchId from the drop-down menu.

- Delete Showroom: An admin can delete a showroom tuple by selecting the showRoomNo and branchID from drop down menus.

- Delete Showing: An admin can delete a showroom tuple by selecting the showRoomNo, branchID, date, and time from drop-down menus.

- Delete Ticket: An admin can delete a Tickets tuple by selecting the TicketID from a drop-down menu.

- Delete System Admin: An admin can delete a System_Admin tuple by selecting the admin's email from a drop-down menu.

- Delete Movie: An admin can delete a Movie tuple by selecting the movie title from a drop-down menu.

- Delete Movie Writer: An admin can delete a Movie_Writer tuple by selecting the writer's name and movie title from drop-down menus.

- Delete Movie Producer: An admin can delete a Movie_Producer tuple by selecting the producer's name and movie title from drop-down menus.

- Delete Cast Member: An admin can delete a Movie_Cast tuple by selecting the Cast member name and movie title from drop-down menus.

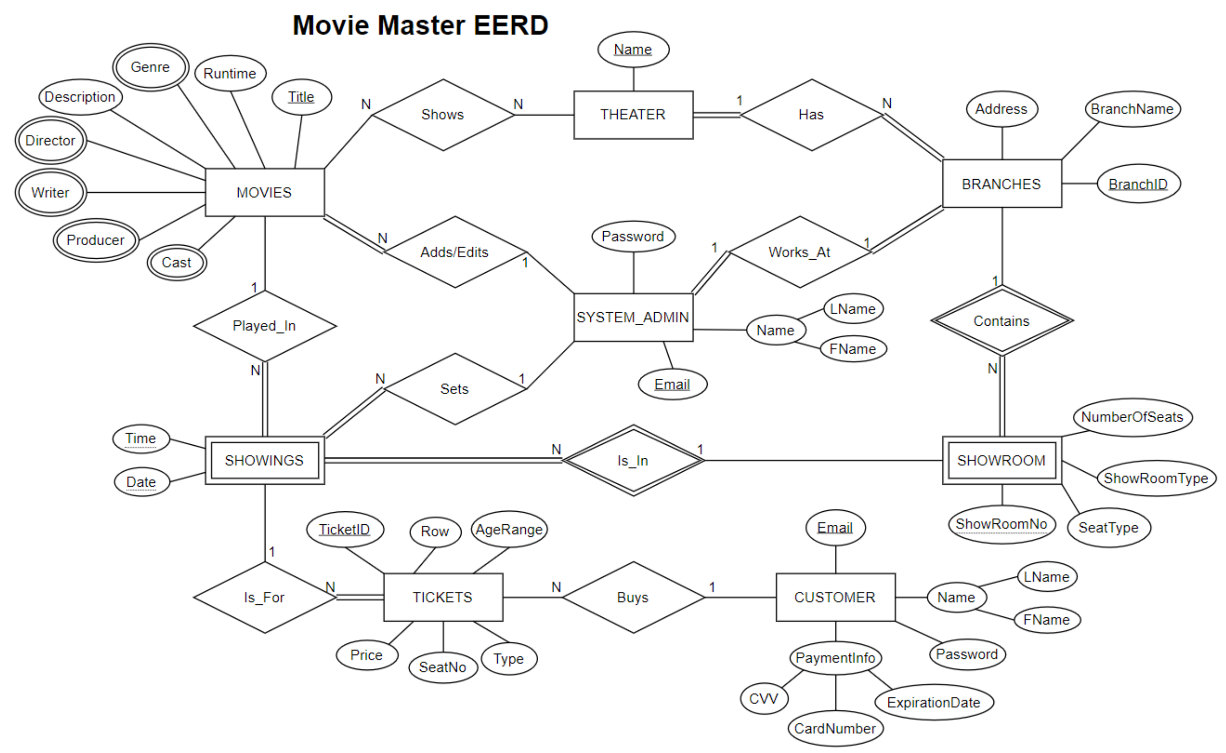
- Delete Movie Genre: An admin can delete a Movie_Genre tuple by selecting the genre and movie title from drop-down menus.

- Delete Movie Director: An admin can delete a Movie_Director tuple by selecting the director name and movie title from drop-down menus.

Note: Admins also have access to all home and ticket select pages. They just cannot checkout as there are no payment details associated with an admin account.

3.3. Entity Relationship Diagram

The below diagram depicts all aspects of the implemented EERD for this project. Note that no design changes were made to this design over the course of the project.



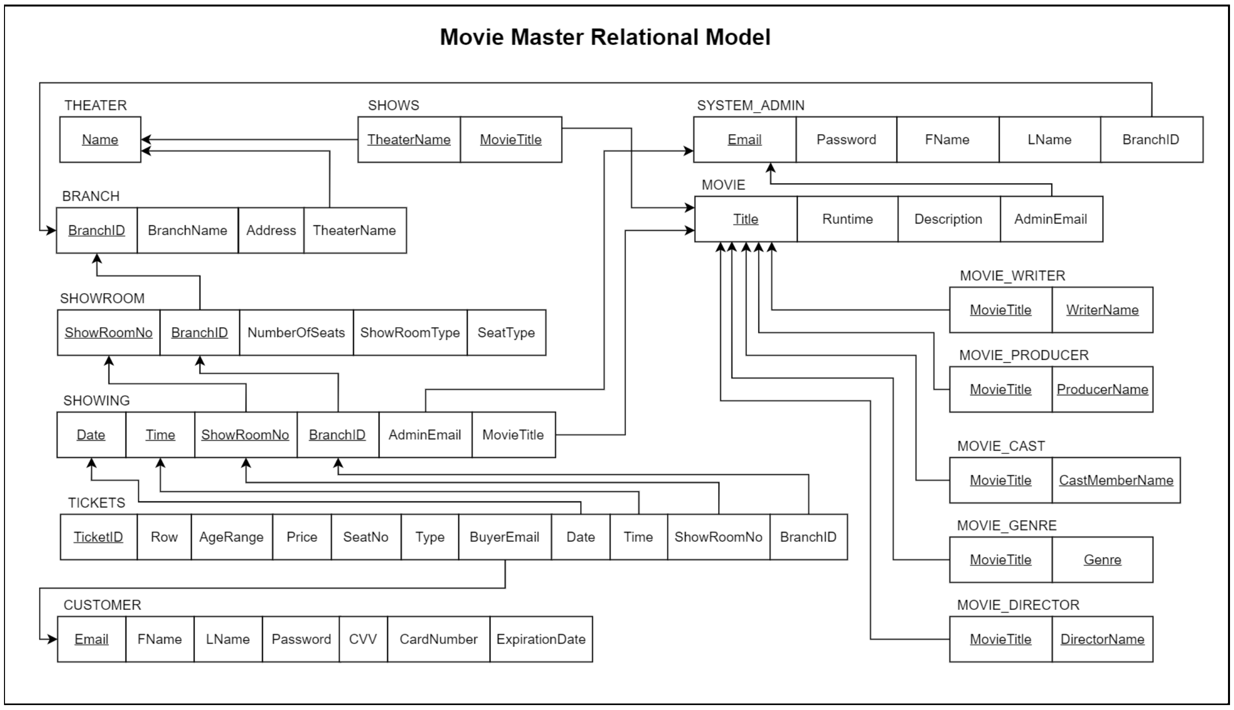
Assumptions:
1. A theater needs at least one branch
2. A branch needs to be associated with a theater
3. A branch needs a system admin
4. A system admin must work at a branch
5. Tickets must be for a showing
6. Movies need to be added by a system admin
7. Showings must be set by a system admin
8. A showing must be in a showroom
9. A showing can only be for one movie
10. A ticket can only be for one showing
11. A theaters name is unique
12. Movie titles are unique

Element Count:
Number of Unique Entities: **8**
Number Weak Entities: **2**
Number of Relationship Types: **10**
Number of End Users: **2**

4. Project Implementation

4.1. Relational Model

The below diagram depicts all aspects of the implemented RM for this project. Note that no significant or unusual decisions were made during the process of deriving this RM from the previously shown EERD.



4.2. Selected DBMS: Frameworks, Languages, and Technologies

The backend of this web application was built in Microsoft Visual Studios 2022 using an ASP.NET core REST API framework. Here, controllers are used to send HTTP requests that allow access to the database that was constructed in Microsoft SQL Server Management Studio 18 (MSSMS). The database was designed with all foreign key and primary key constraints as shown in the relational model as well as with the appropriate cascade delete functionality applied. MSSMS was also used to write and generate the stored procedures used to prevent SQL injection and to create specialized queries. The frontend interface was designed using Angular in Visual Studio Code. The Angular Materials library was also imported to give access to specialty components used throughout the website. Languages used for the application include typescript, CSS, HTML, and JSON for the frontend, as well as C# for the backend.

4.3. SQL Statements for Implemented Transactions

4.3.1 Admin Login

```
SELECT *
FROM System_Admin
WHERE System_Admin.email = @email
```

4.3.2 Get Tickets

```
SELECT *
FROM Tickets
WHERE Tickets.Date = @date AND
        Tickets.Time = @time AND
        Tickets.ShowRoomNo = @showRoomNo AND
        Tickets.BranchID = @branchID AND
        Tickets.BuyerEmail = '';
```

4.3.3 Last Tickets

```
SELECT *
FROM Tickets AS T
WHERE T.TicketID=(
   SELECT max(TicketID) FROM Tickets)
```

4.3.4 Search City

```
SELECT * FROM Showing AS S WHERE
   EXISTS (SELECT * FROM Showroom AS SR, Branch AS B
         WHERE SR.ShowRoomNo = S.ShowRoomNo
            AND SR.BranchID = S.BranchID
            AND S.BranchID = B.BranchID
            AND CHARINDEX(@val, B.Address) > 0)
```

4.3.5 Search Movie Cast

```
SELECT * FROM MOVIE_CAST AS M
WHERE M.MovieTitle = @val
```

4.3.6 Search Movie Director

```
SELECT * FROM MOVIE_DIRECTOR AS M
WHERE M.MovieTitle = @val
```

4.3.7 Search Movie Genre

```
SELECT * FROM MOVIE_GENRE AS M
```

```
WHERE M.MovieTitle = @val
```

4.3.8 Search Movie Producer

```
SELECT * FROM MOVIE_PRODUCER AS M
WHERE M.MovieTitle = @val
```

4.3.9 Search Movie Writer

```
SELECT * FROM MOVIE_WRITER AS M
WHERE M.MovieTitle = @val
```

4.3.10 Search Theater

```
SELECT * FROM Showing AS S WHERE
   EXISTS (SELECT * FROM Showroom AS SR, Branch AS B
         WHERE SR.ShowRoomNo = S.ShowRoomNo
            AND SR.BranchID = S.BranchID
            AND S.BranchID = B.BranchID
            AND B.BranchID = @val)
```

4.3.11 Get All Theaters

```
SELECT *
FROM Theater;
```

4.3.12 Get All Branches

```
SELECT *
FROM Branch;
```

4.3.13 Get All Showrooms

```
SELECT *
FROM Showroom;
```

4.3.14 Get All Showings

```
SELECT *
FROM Showing;
```

4.3.15 Get All Customers

```
SELECT *
FROM Customer;
```

4.3.16 Get All System Admins
```
SELECT *
FROM System_Admin;
```

4.3.17 Get All Movies
```
SELECT *
FROM Movie;
```

4.3.18 Get All Movie Writers
```
SELECT *
FROM Movie_Writer;
```

4.3.19 Get All Movie Producers
```
SELECT *
FROM Movie_Producer;
```

4.3.20 Get All Movie Casts
```
SELECT *
FROM Movie_Cast;
```

4.3.21 Get All Movie Genres
```
SELECT *
FROM Movie_Genre;
```

4.3.22 Get All Movie Directors
```
SELECT *
FROM Movie_Director;
```

4.3.23 Get All Shows
```
SELECT *
FROM Shows;
```

4.3.24 Get All Tickets
```
SELECT *
FROM Tickets;
```

4.3.25 Delete Theater
```
DELETE
FROM Theater
```

WHERE Name = @Name;

4.3.26 Delete Branch

```
DELETE
FROM Branch
WHERE BranchID = @BranchID;
```

4.3.27 Delete Showroom

```
DELETE
FROM Showroom
WHERE ShowRoomNo = @ShowRoomNo;
```

4.3.28 Delete Showing

```
DELETE
FROM Showing
WHERE Date = @Date AND
Time = @Time AND
ShowRoomNo = @ShowRoomNo AND
BranchID = @BranchID;
```

4.3.29 Delete Tickets

```
DELETE
FROM Tickets
WHERE TicketID = @TicketID;
```

4.3.30 Delete Customer

```
DELETE
FROM Customer
WHERE Email = @Email;
```

4.3.31 Delete Shows

```
DELETE
FROM Shows
WHERE TheaterName = @TheaterName AND
MovieTitle = @MovieTitle;
```

4.3.32 Delete System_Admin

```
DELETE
FROM System_Admin
```

```
WHERE Email = @Email;
```

4.3.33 Delete Movie

```
DELETE
FROM Movie
WHERE Title = @Title
MovieTitle = @MovieTitle;
```

4.3.34 Delete Movie_Writer

```
DELETE
FROM Movie_Writer
WHERE WriterName = @WriterName AND
MovieTitle = @MovieTitle;
```

4.3.35 Delete Movie_Producer

```
DELETE
FROM Movie_Producer
WHERE ProducerName = @ProducerName AND
MovieTitle = @MovieTitle;
```

4.3.36 Delete Movie_Cast

```
DELETE
FROM Movie_Cast
WHERE CastMemberName = @CastMemberName AND
MovieTitle = @MovieTitle;
```

4.3.37 Delete Movie_Genre

```
DELETE
FROM Movie_Genre
WHERE Genre = @Genre AND
MovieTitle = @MovieTitle;
```

4.3.38 Delete Movie_Director

```
DELETE
FROM Shows
WHERE DirectorName = @DirectorName AND
MovieTitle = @MovieTitle;
```

4.3.39 Signup

INSERT INTO Customer
values(@Email, @FName, @LNAME, @Password, @CVV, @CardNumber, @ExpirationDate);

4.3.40 Insert Theater

INSERT INTO Theater
values(@Name);

4.3.41 Insert Branch

INSERT INTO Branch
values(@BranchID, @BranchName, @Address, @TheaterName);

4.3.42 Insert Showroom

INSERT INTO Showroom
values(@ShowRoomNo, @BranchID, @NumberOfSeats, @ShowRoomType, @SeatType);

4.3.43 Insert Showing

INSERT INTO Showing
values(@Date, @Time, @ShowRoomNo, @BranchID, @AdminEmail, @MovieTitle);

4.3.44 Insert Tickets

INSERT INTO Tickets
values(@TicketID, @Row, @AgeRange, @Price, @SeatNo, @Type, @BuyerEmail, @Date, @Time, @ShowRoomNo, @BranchID);

4.3.45 Insert Customer

INSERT INTO Customer
values(@Email, @FName, @LName, @Password, @CVV, @CardNumber, @ExpirationDate);

4.3.46 Insert Shows

INSERT INTO Shows
values(@TheaterName, @MovieTitle);

4.3.47 Insert System_Admin

INSERT INTO System_Admin

values(@Email, @Password, @FName, @LName, @BranchID);

4.3.48 Insert Movie

INSERT INTO Movie
values(@Title, @Runtime, @Description, @AdminEmail);

4.3.49 Insert Movie_Writer

INSERT INTO Movie_Writer
values(@MovieTitle, @WriterName);

4.3.50 Insert Movie_Producer

INSERT INTO Movie_Producer
values(@MovieTitle, @WriterProducer);

4.3.51 Insert Movie_Cast

INSERT INTO Movie_Cast
values(@MovieTitle, @CastMemberName);

4.3.52 Insert Movie_Genre

INSERT INTO Movie_Writer
values(@MovieTitle, @Genre);

4.3.53 Insert Movie_Director

INSERT INTO Movie_Writer
values(@MovieTitle, @DirectorName);

5. API Documentation (Generated Using Postman)

The following link provides public access to our generated API documentation. All 53 endpoints are depicted, including the endpoint URLs, a description, a JSON body where applicable, as well as a saved example showing the recorded output of executing the request. API documentation can be accessed here:
https://documenter.getpostman.com/view/20379496/Uyr4Jeqn

6. User Guide

6.1. System Setup and How to Run

The front end requires a 32-bit windows version of NodeJS version 16.13.1. To install a node modules folder you will want to run the command npm install once navigated to the installed project folder. Then you can enter npm start.

To run the application the application files first must be downloaded off of the github repository located here, https://github.com/JaredAssen/MovieMaster. This can be done most easily by simply cloning or forking the repository, and then navigating to the main directory (MovieMaster) before running the program. Once the files are downloaded run "ng serve" to start the frontend. The user can then navigate to http://localhost:4200/ to view the website.

The database was constructed using Microsoft SQL Management Studio. A .sql script file is provided in the git repository at https://github.com/JaredAssen/MovieMaster/blob/main/MovieMasterScript.sql. This file can be used to generate the database along with all the stored procedures used.

Finally, the API was created using the .NET 6.0 Web API framework in Visual Studios 2022. To run this, open the MovieMasterAPI.sin file located in the MovieMasterAPI folder. You may have to go to the AppSettings.json file and change the DevConnection string so it contains the connection string for your SQL server. Then run the solution. A web page will open up at http://localhost:5167. Once all this is done the web application can be used as intended.

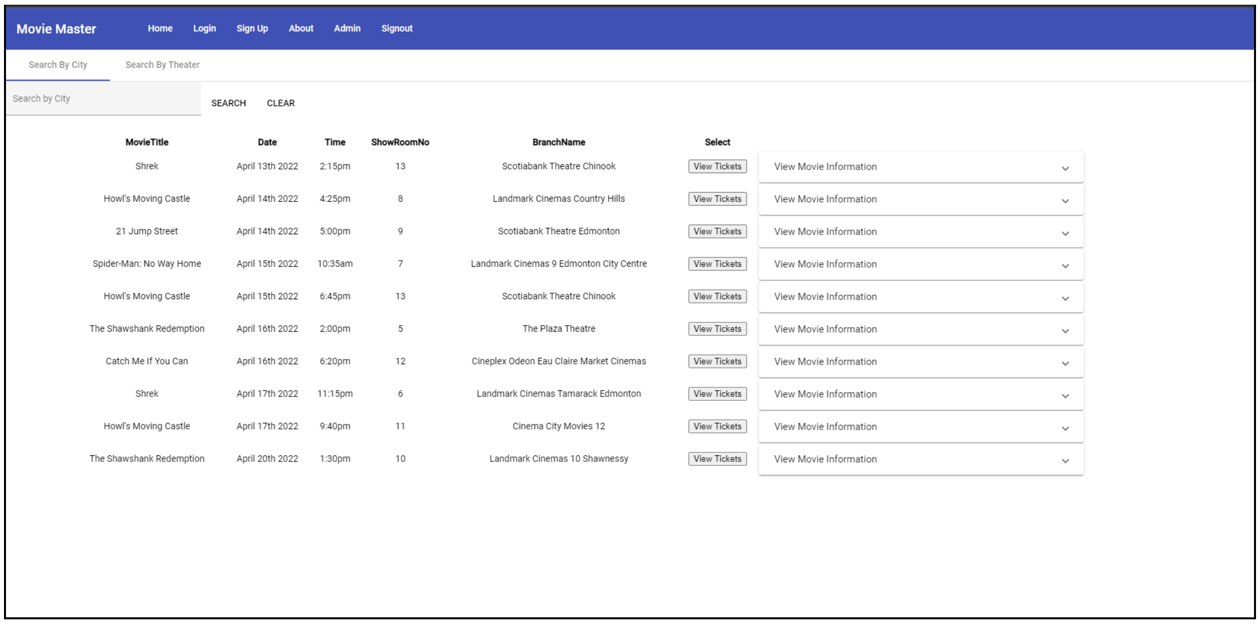
6.2. Guide for General Users

The general user will begin on the website's home page as does every user. From here, they will be able to see a table of all current showings in the database as well as drop-down menus that contain more information on the movie that each showing is for [Figure 6.2.1]. Upon clicking on a dropdown, the user will be shown the movie titles, runtime, description, cast member names, directors, genres, producers, and writers [Figure 6.2.2]. Located at the top of the page will be a search bar to search for showings by city name. Users can input a city and hit the search button to see if there are any showings there. They can also hit the clear button to clear any search filters previously applied. This feature uses a stored procedure to query the database for all showings that have an address that contains the inputted city in it [Figure 6.2.3]. There is also a search by theater and branch option in the scroll bar near the top of the page that users can press to scroll over to a search by theater and branch name drop-down. This drop-down will contain a list of all theater names and their associated branch names in the database for users to select as a filter for the showings list [Figure 6.2.4].

From here, users can select to view tickets for whichever showing they are interested in. Upon clicking the view tickets button beside a showing, all the ticket information for that showing that is stored in the database will be displayed in a table [Figure 6.2.5]. Users can then press a button beside each ticket to add it to their cart. The next step would be to continue to checkout, but if a user is not logged in to the system yet they will be taken to the login page after hitting the checkout button. A user can then input their email and password to log in. The system will check the input against all stored customer emails and passwords and will only log them in once a match is found [Figure 6.2.6]. If a user does not yet have an account they will need to hit the sign up button in the top toolbar to create an account. Here, the user will be required to enter contact and payment information which will be used for checkout and login purposes [Figure

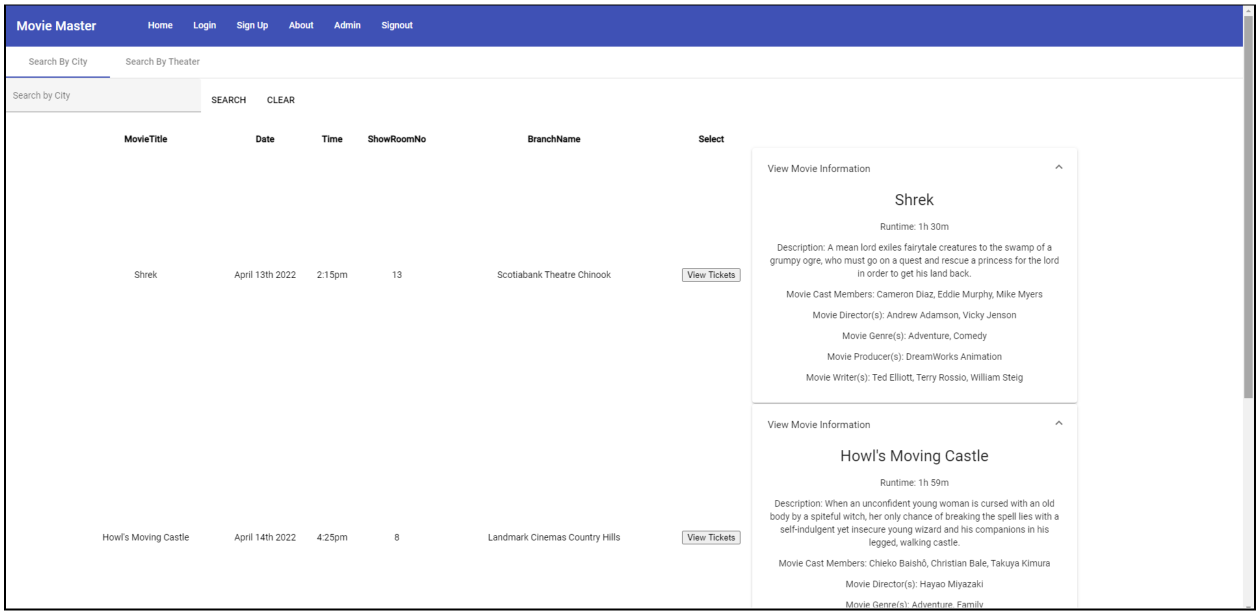
6.2.7]. Hitting submit on the signup page with the appropriate information inputted will take the user back to the login page to log in. Since the customer's email is the primary key in the database, no two accounts can use the same email. Now that a customer is logged in, they can navigate back through the ticket select pages and will now have access to the checkout page. The checkout page will show a tabular summary of all the tickets in a customer's cart. It will also display the total price of the tickets and the logged-in user's payment information [Figure 6.2.8]. From here, users can confirm their purchase or clear their cart, which will display the purchase success page [Figure 6.2.9]. If a user is done with the website, they can sign out of their account using the sign-out button in the top toolbar.

Users will also be able to view an about page by clicking a button in the upper toolbar. This will direct them to a web page that has a short description of the application [Figure 6.2.10].



MovieTitle	Date	Time	ShowRoomNo	BranchName	Select	
Shrek	April 13th 2022	2:15pm	13	Scotiabank Theatre Chinook	View Tickets	View Movie Information
Howl's Moving Castle	April 14th 2022	4:25pm	8	Landmark Cinemas Country Hills	View Tickets	View Movie Information
21 Jump Street	April 14th 2022	5:00pm	9	Scotiabank Theatre Edmonton	View Tickets	View Movie Information
Spider-Man: No Way Home	April 15th 2022	10:35am	7	Landmark Cinemas 9 Edmonton City Centre	View Tickets	View Movie Information
Howl's Moving Castle	April 15th 2022	6:45pm	13	Scotiabank Theatre Chinook	View Tickets	View Movie Information
The Shawshank Redemption	April 16th 2022	2:00pm	5	The Plaza Theatre	View Tickets	View Movie Information
Catch Me If You Can	April 16th 2022	6:20pm	12	Cineplex Odeon Eau Claire Market Cinemas	View Tickets	View Movie Information
Shrek	April 17th 2022	11:15pm	6	Landmark Cinemas Tamarack Edmonton	View Tickets	View Movie Information
Howl's Moving Castle	April 17th 2022	9:40pm	11	Cinema City Movies 12	View Tickets	View Movie Information
The Shawshank Redemption	April 20th 2022	1:30pm	10	Landmark Cinemas 10 Shawnessy	View Tickets	View Movie Information

Figure 6.2.1: Movie Master Home Page



MovieTitle	Date	Time	ShowRoomNo	BranchName	Select
Shrek	April 13th 2022	2:15pm	13	Scotiabank Theatre Chinook	View Tickets
Howl's Moving Castle	April 14th 2022	4:25pm	8	Landmark Cinemas Country Hills	View Tickets

View Movie Information

Shrek

Runtime: 1h 30m

Description: A mean lord exiles fairytale creatures to the swamp of a grumpy ogre, who must go on a quest and rescue a princess for the lord in order to get his land back.

Movie Cast Members: Cameron Diaz, Eddie Murphy, Mike Myers

Movie Director(s): Andrew Adamson, Vicky Jenson

Movie Genre(s): Adventure, Comedy

Movie Producer(s): DreamWorks Animation

Movie Writer(s): Ted Elliott, Terry Rossio, William Steig

View Movie Information

Howl's Moving Castle

Runtime: 1h 59m

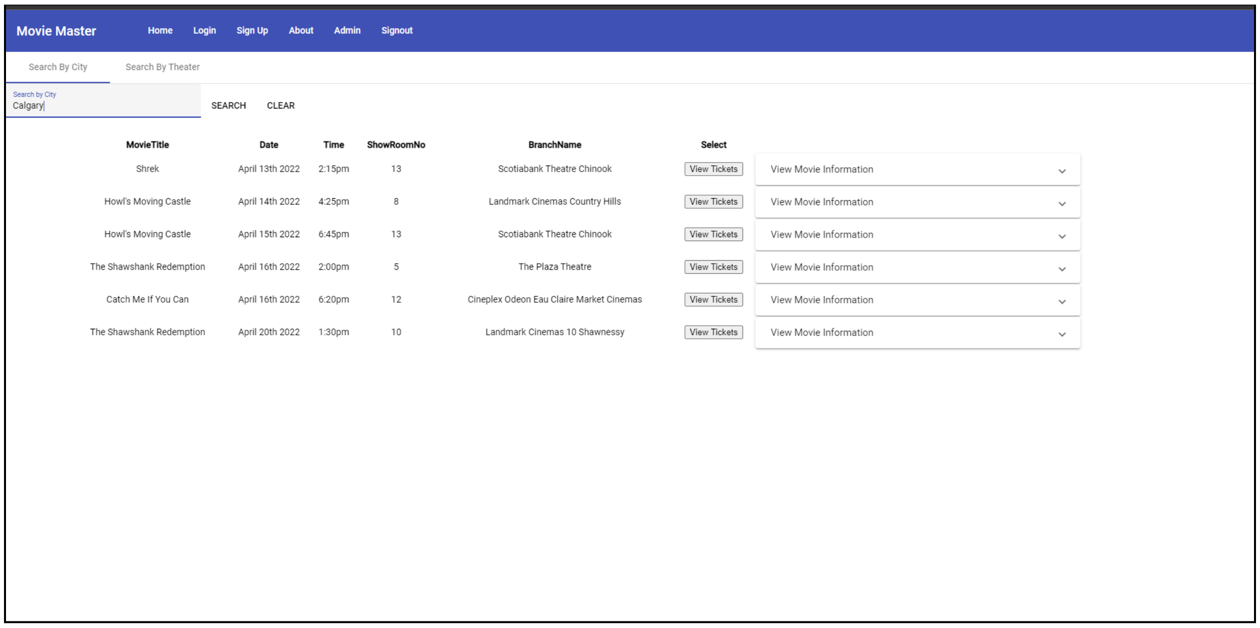
Description: When an unconfident young woman is cursed with an old body by a spiteful witch, her only chance of breaking the spell lies with a self-indulgent yet insecure young wizard and his companions in his legged, walking castle.

Movie Cast Members: Chieko Baishō, Christian Bale, Takuya Kimura

Movie Director(s): Hayao Miyazaki

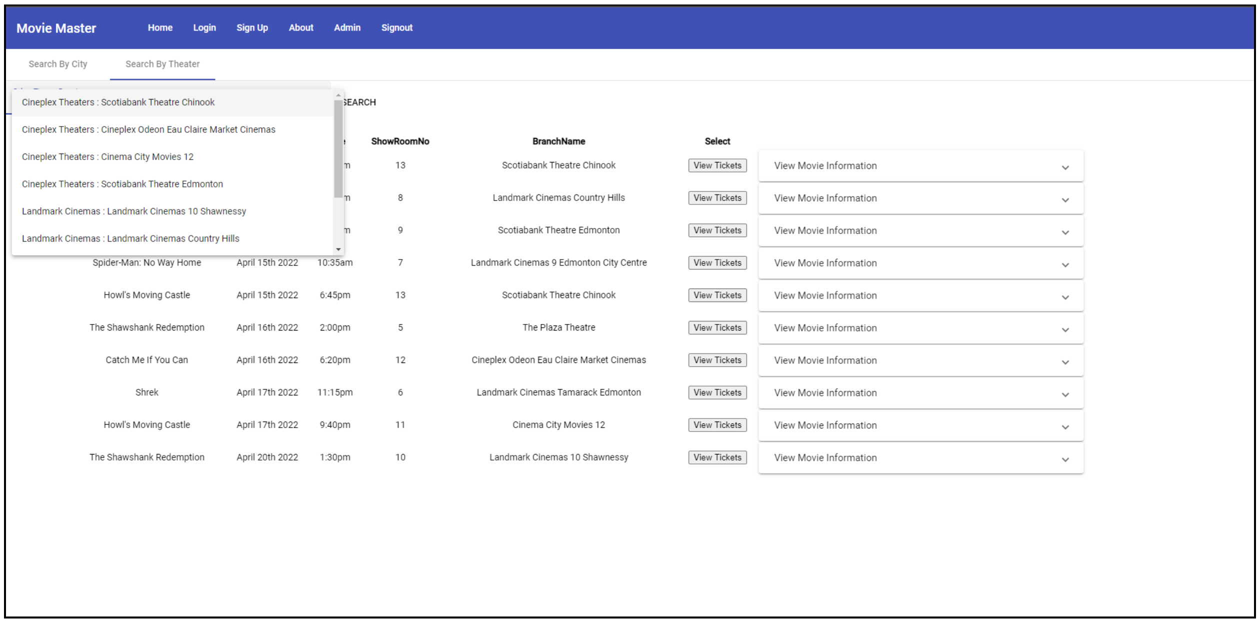
Movie Genre(s): Adventure, Family

Figure 6.2.2: Drop-downs Showing Movie Information



MovieTitle	Date	Time	ShowRoomNo	BranchName	Select	
Shrek	April 13th 2022	2:15pm	13	Scotiabank Theatre Chinook	View Tickets	View Movie Information
Howl's Moving Castle	April 14th 2022	4:25pm	8	Landmark Cinemas Country Hills	View Tickets	View Movie Information
Howl's Moving Castle	April 15th 2022	6:45pm	13	Scotiabank Theatre Chinook	View Tickets	View Movie Information
The Shawshank Redemption	April 16th 2022	2:00pm	5	The Plaza Theatre	View Tickets	View Movie Information
Catch Me If You Can	April 16th 2022	6:20pm	12	Cineplex Odeon Eau Claire Market Cinemas	View Tickets	View Movie Information
The Shawshank Redemption	April 20th 2022	1:30pm	10	Landmark Cinemas 10 Shawnessy	View Tickets	View Movie Information

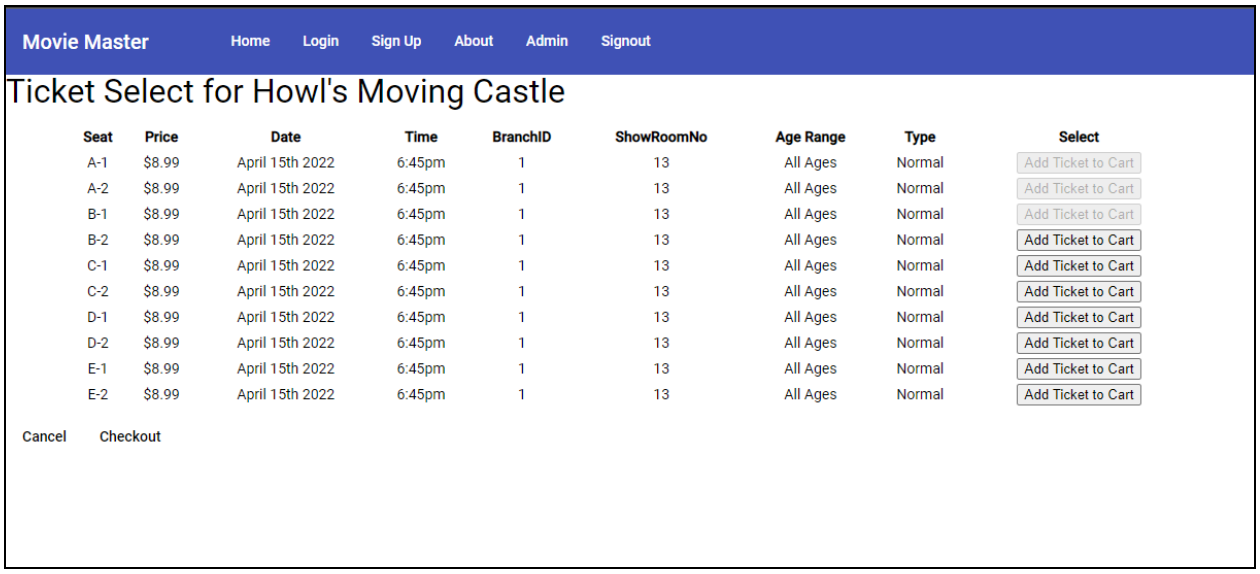
Figure 6.2.3: Searching by City Name



Cineplex Theaters : Scotiabank Theatre Chinook
Cineplex Theaters : Cineplex Odeon Eau Claire Market Cinemas
Cineplex Theaters : Cinema City Movies 12
Cineplex Theaters : Scotiabank Theatre Edmonton
Landmark Cinemas : Landmark Cinemas 10 Shawnessy
Landmark Cinemas : Landmark Cinemas Country Hills

			ShowRoomNo	BranchName	Select	
		m	13	Scotiabank Theatre Chinook	View Tickets	View Movie Information
		m	8	Landmark Cinemas Country Hills	View Tickets	View Movie Information
		m	9	Scotiabank Theatre Edmonton	View Tickets	View Movie Information
Spider-Man: No Way Home	April 15th 2022	10:35am	7	Landmark Cinemas 9 Edmonton City Centre	View Tickets	View Movie Information
Howl's Moving Castle	April 15th 2022	6:45pm	13	Scotiabank Theatre Chinook	View Tickets	View Movie Information
The Shawshank Redemption	April 16th 2022	2:00pm	5	The Plaza Theatre	View Tickets	View Movie Information
Catch Me If You Can	April 16th 2022	6:20pm	12	Cineplex Odeon Eau Claire Market Cinemas	View Tickets	View Movie Information
Shrek	April 17th 2022	11:15pm	6	Landmark Cinemas Tamarack Edmonton	View Tickets	View Movie Information
Howl's Moving Castle	April 17th 2022	9:40pm	11	Cinema City Movies 12	View Tickets	View Movie Information
The Shawshank Redemption	April 20th 2022	1:30pm	10	Landmark Cinemas 10 Shawnessy	View Tickets	View Movie Information

Figure 6.2.4: Searching by Theater and Branch Name



Ticket Select for Howl's Moving Castle

Seat	Price	Date	Time	BranchID	ShowRoomNo	Age Range	Type	Select
A-1	$8.99	April 15th 2022	6:45pm	1	13	All Ages	Normal	Add Ticket to Cart
A-2	$8.99	April 15th 2022	6:45pm	1	13	All Ages	Normal	Add Ticket to Cart
B-1	$8.99	April 15th 2022	6:45pm	1	13	All Ages	Normal	Add Ticket to Cart
B-2	$8.99	April 15th 2022	6:45pm	1	13	All Ages	Normal	Add Ticket to Cart
C-1	$8.99	April 15th 2022	6:45pm	1	13	All Ages	Normal	Add Ticket to Cart
C-2	$8.99	April 15th 2022	6:45pm	1	13	All Ages	Normal	Add Ticket to Cart
D-1	$8.99	April 15th 2022	6:45pm	1	13	All Ages	Normal	Add Ticket to Cart
D-2	$8.99	April 15th 2022	6:45pm	1	13	All Ages	Normal	Add Ticket to Cart
E-1	$8.99	April 15th 2022	6:45pm	1	13	All Ages	Normal	Add Ticket to Cart
E-2	$8.99	April 15th 2022	6:45pm	1	13	All Ages	Normal	Add Ticket to Cart

Figure 6.2.5: Ticket Selection Page

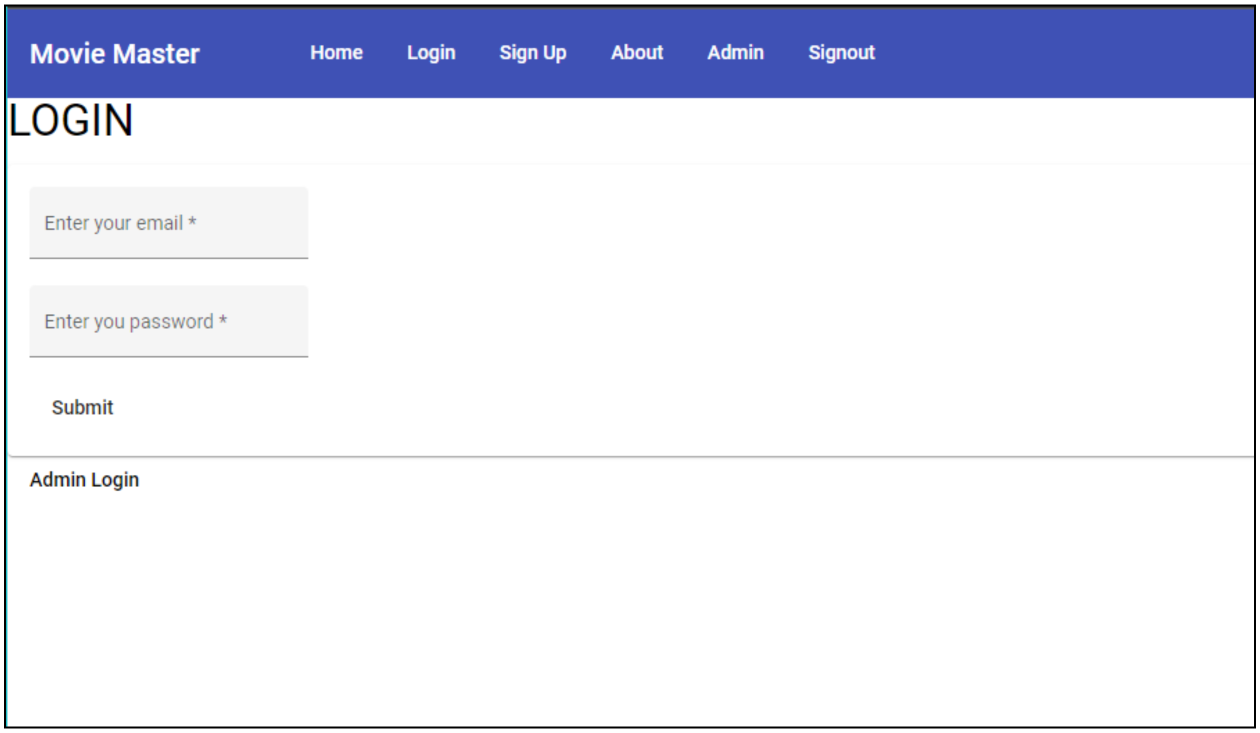


Figure 6.2.6: User Login Page

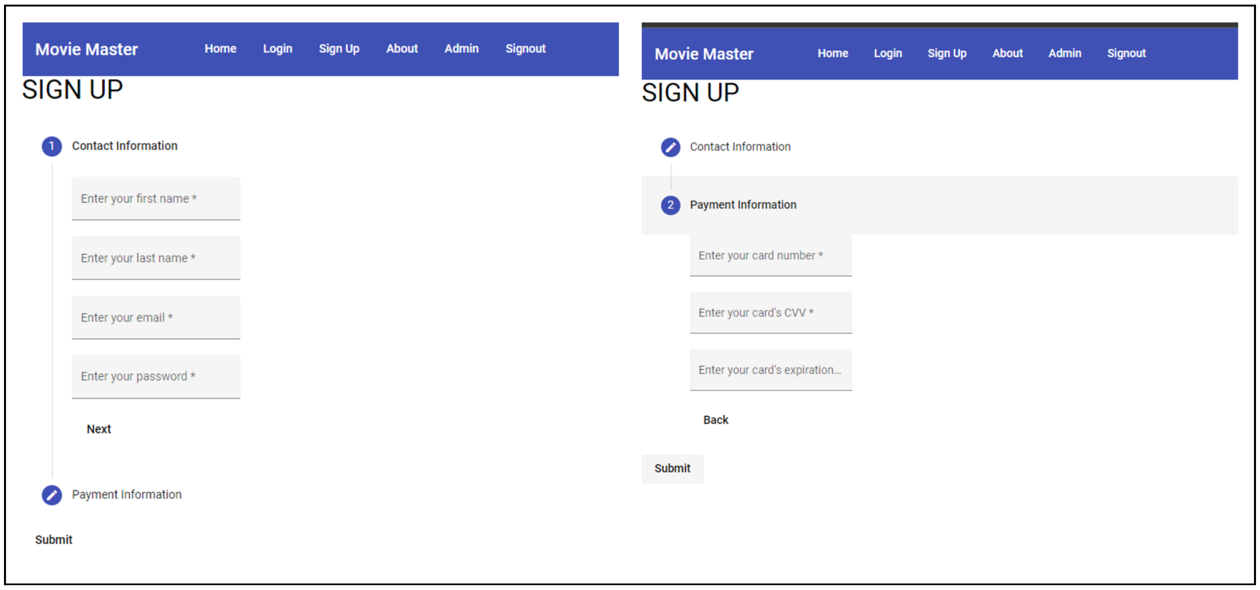
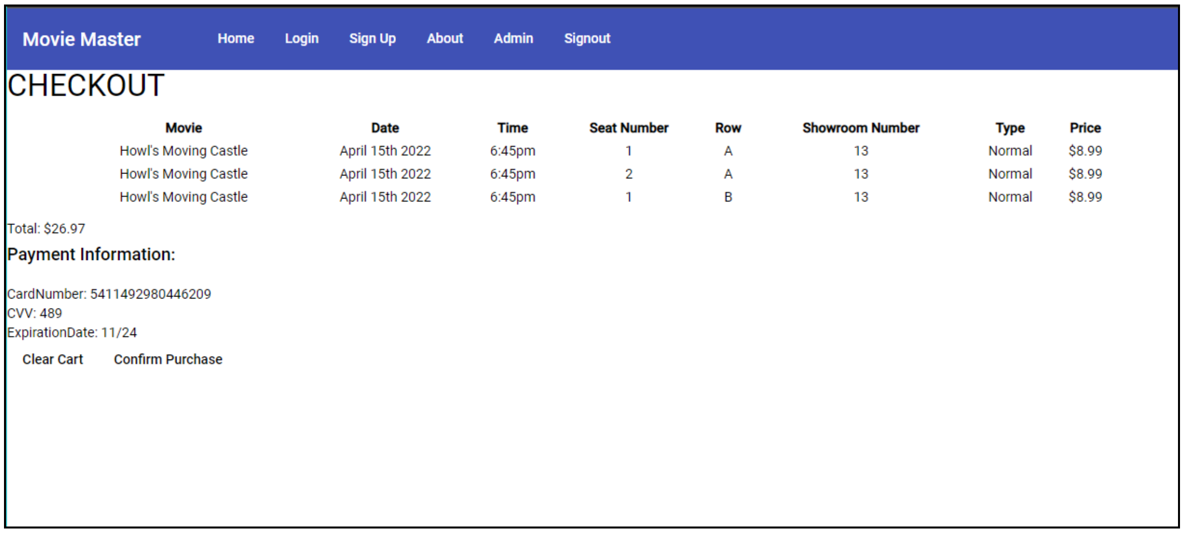


Figure 6.2.7: User Sign Up Page



CHECKOUT

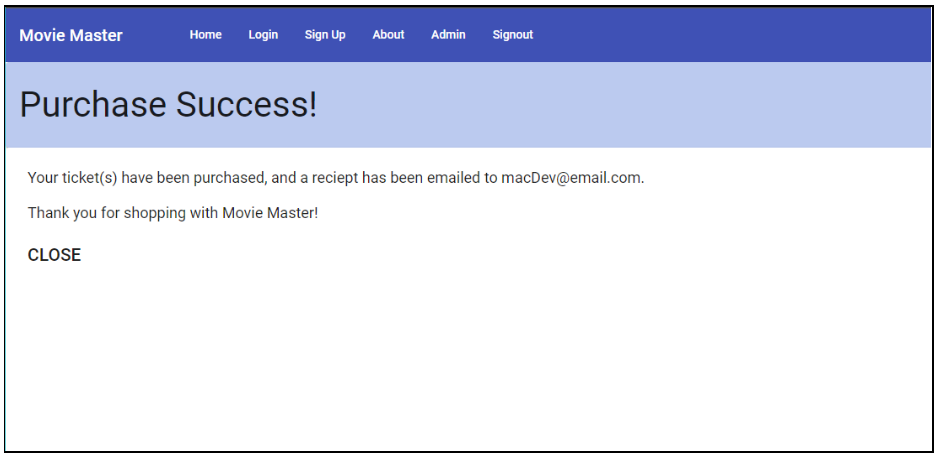
Movie	Date	Time	Seat Number	Row	Showroom Number	Type	Price
Howl's Moving Castle	April 15th 2022	6:45pm	1	A	13	Normal	$8.99
Howl's Moving Castle	April 15th 2022	6:45pm	2	A	13	Normal	$8.99
Howl's Moving Castle	April 15th 2022	6:45pm	1	B	13	Normal	$8.99

Total: $26.97

Payment Information:

CardNumber: 5411492980446209
CVV: 489
ExpirationDate: 11/24

Figure 6.2.8: Checkout Page

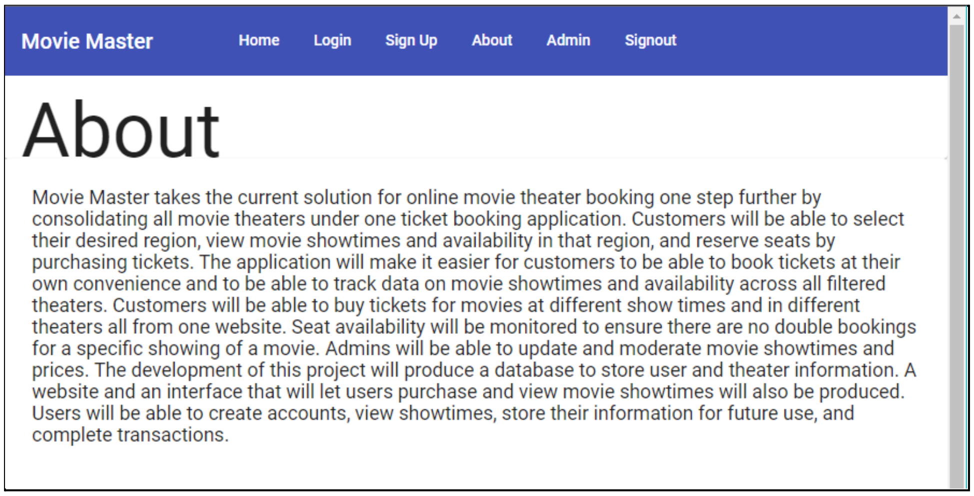


Purchase Success!

Your ticket(s) have been purchased, and a reciept has been emailed to macDev@email.com.

Thank you for shopping with Movie Master!

Figure 6.2.9: Purchase Success Page



About

Movie Master takes the current solution for online movie theater booking one step further by consolidating all movie theaters under one ticket booking application. Customers will be able to select their desired region, view movie showtimes and availability in that region, and reserve seats by purchasing tickets. The application will make it easier for customers to be able to book tickets at their own convenience and to be able to track data on movie showtimes and availability across all filtered theaters. Customers will be able to buy tickets for movies at different show times and in different theaters all from one website. Seat availability will be monitored to ensure there are no double bookings for a specific showing of a movie. Admins will be able to update and moderate movie showtimes and prices. The development of this project will produce a database to store user and theater information. A website and an interface that will let users purchase and view movie showtimes will also be produced. Users will be able to create accounts, view showtimes, store their information for future use, and complete transactions.

Figure 6.2.9: About Page

6.3. Guide for System Administrators

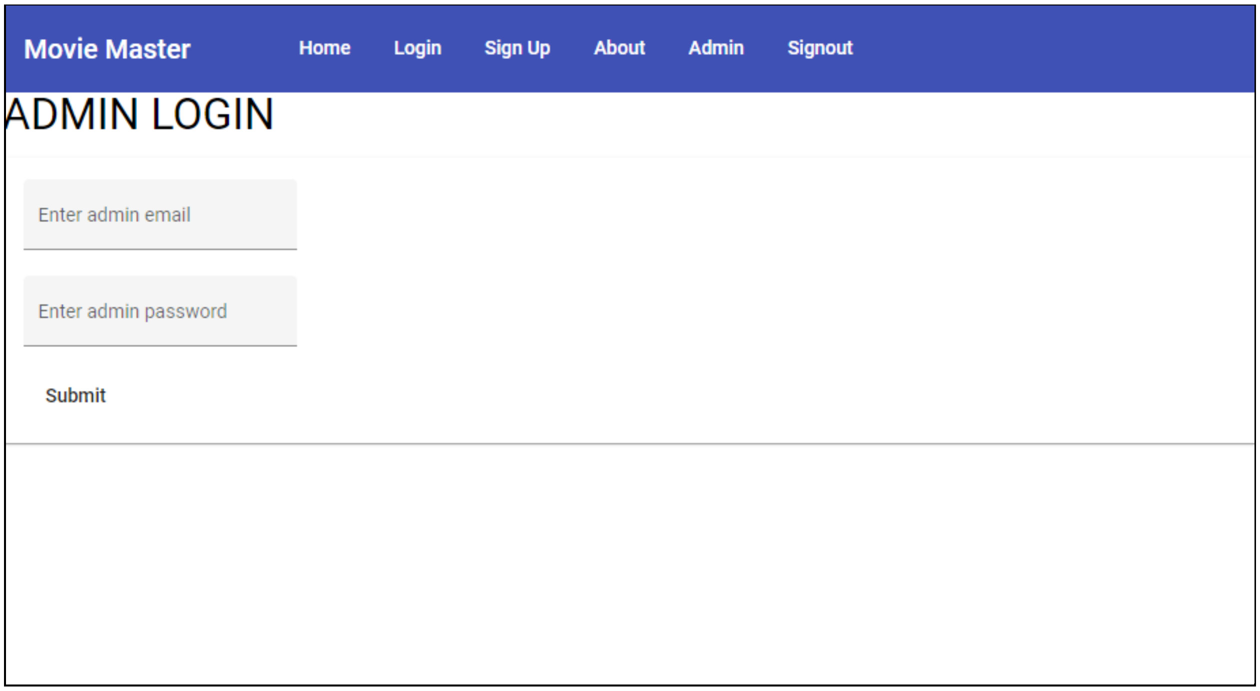
System administrators gain special access to various elements of the system, which permit them to manually alter all data in the Movie Master database. In addition to all components available to general users as outlined in section 6.2, system administrators can navigate to the admin login page by selecting the admin button in the top navigation bar or choosing the admin login option visible on the general login page. From here, administrators can enter their unique emails and passwords as are stored in the database, and click submit to login [Figure 6.3.1].

Note that for security reasons, a system admin can only be added through the UI by one that already exists. This means that on system setup an initial admin will need to be added manually, which can be done by sending a POST request to the *http://localhost:5167/api/System_Admin* endpoint, with a body similar to the one provided here:

```
{
  "email": "admin@email.com",
  "password": "pass",
  "fName": "Admin",
  "lName": "Istrator",
  "branchID": 2
}
```

This can also be accessed through the <<System Admins>> / Add System Admin endpoint provided in our API documentation. Additionally, the provided database state that can be imported also contains two entries for current system admins.

Once logged into the system, system admins are able to access the admin page, which displays two side-by-side columns of add and delete buttons, which can be switched between through the use of a sliding Add/Delete menu bar near the top of the page [Figure 6.3.2]. Each of the displayed options navigates to individual UI pages that allow the system admin to enter all data needed to create a new tuple in the database or delete an existing one. Note that all entries which require a selection of information currently contained within the database are implemented through the use of a dropdown menu. The “Add” pages that are provided include the ability to add a theater, shows, branch, showroom, showing, system admin, and movie. Additionally, any selected number of tickets can be generated using the add tickets page, which automatically assigns row and seat numbers based on the number of rows entered [Figure 6.3.3]. The “Delete” pages work similarly providing functionality for system admins to select and delete any theater, shows, branch, showroom, showing, system admin, movie, ticket, or customer based on the primary key [Figure 6.3.4]. Finally, system admins are also provided with the option to add or delete any of the multivalued attributes associated with a movie, which include writers, producers, directors, genres, and cast members [Figure 6.3.5]. Note that these options are outlined in detail in section 3.2, and are displayed in the associated figures below.



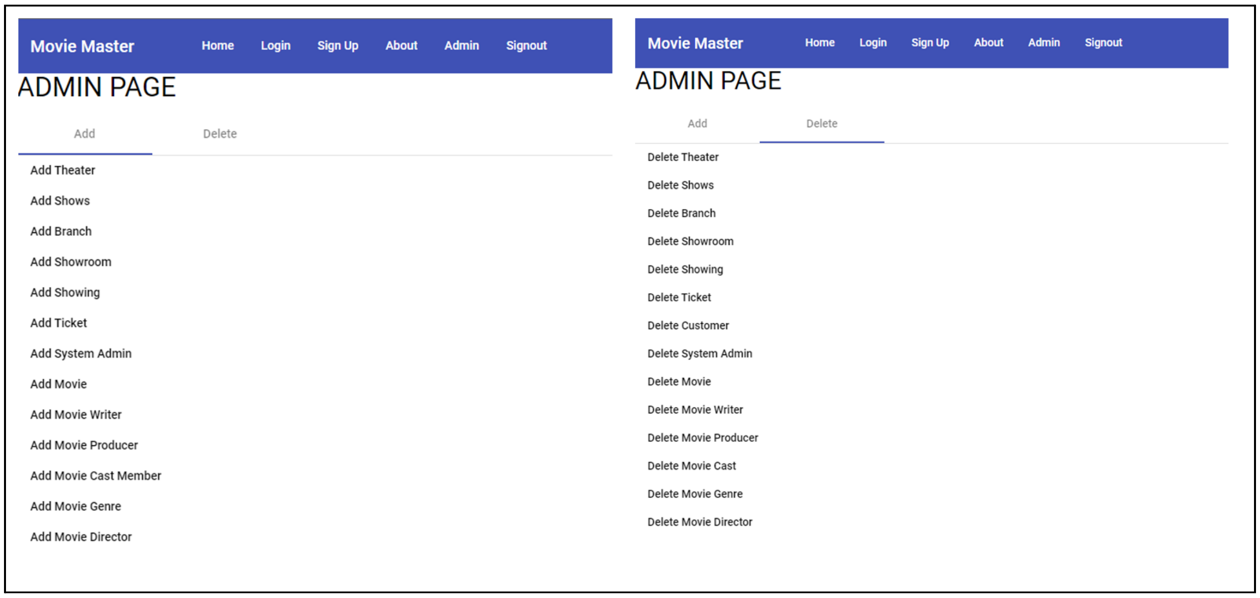
ADMIN LOGIN

Enter admin email

Enter admin password

Submit

Figure 6.3.1: Admin Login Page



ADMIN PAGE

Add | Delete

Add Theater
Add Shows
Add Branch
Add Showroom
Add Showing
Add Ticket
Add System Admin
Add Movie
Add Movie Writer
Add Movie Producer
Add Movie Cast Member
Add Movie Genre
Add Movie Director

ADMIN PAGE

Add | Delete

Delete Theater
Delete Shows
Delete Branch
Delete Showroom
Delete Showing
Delete Ticket
Delete Customer
Delete System Admin
Delete Movie
Delete Movie Writer
Delete Movie Producer
Delete Movie Cast
Delete Movie Genre
Delete Movie Director

Figure 6.3.2: Admin Main Page(s)

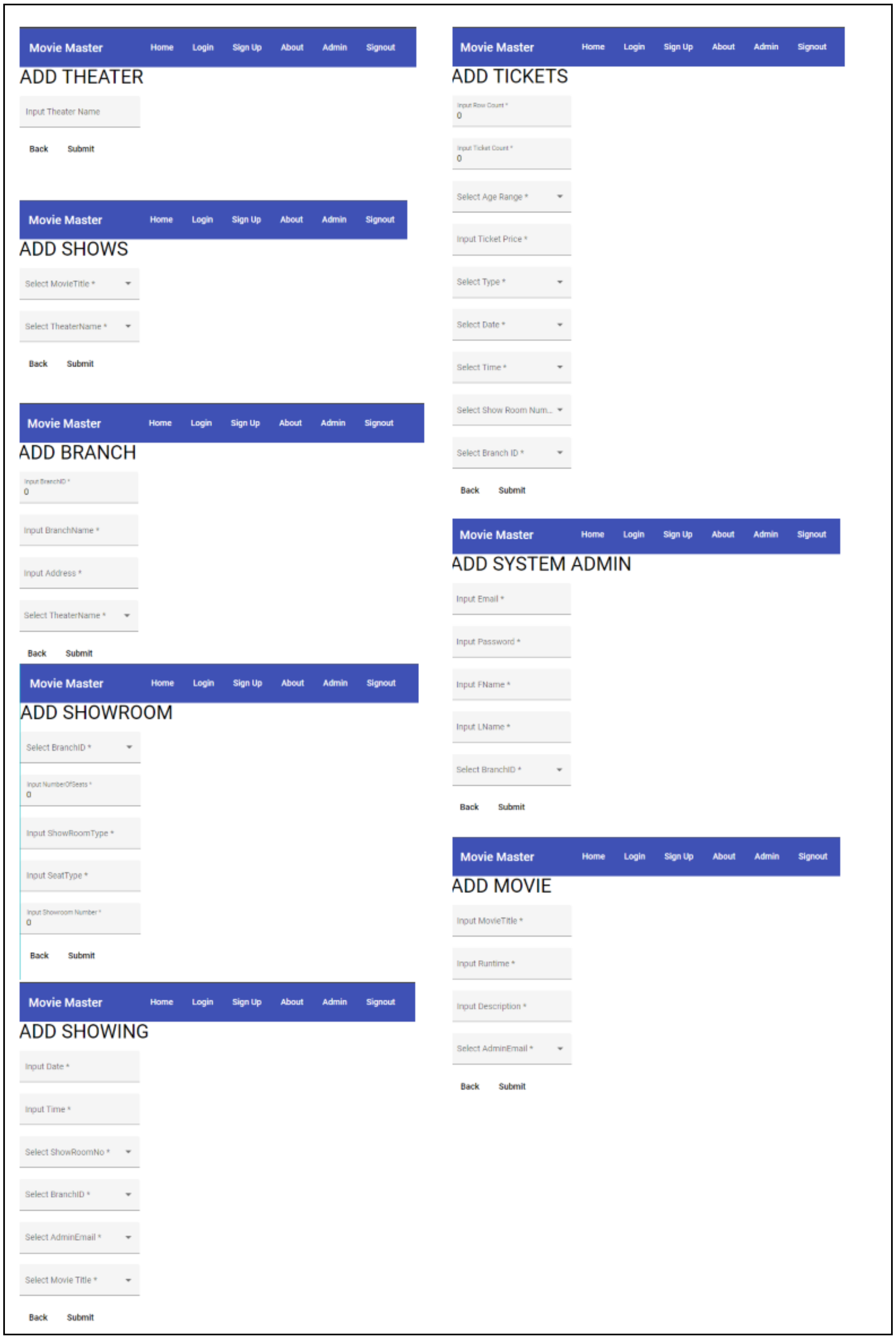


Figure 6.3.3: Admin Add Pages

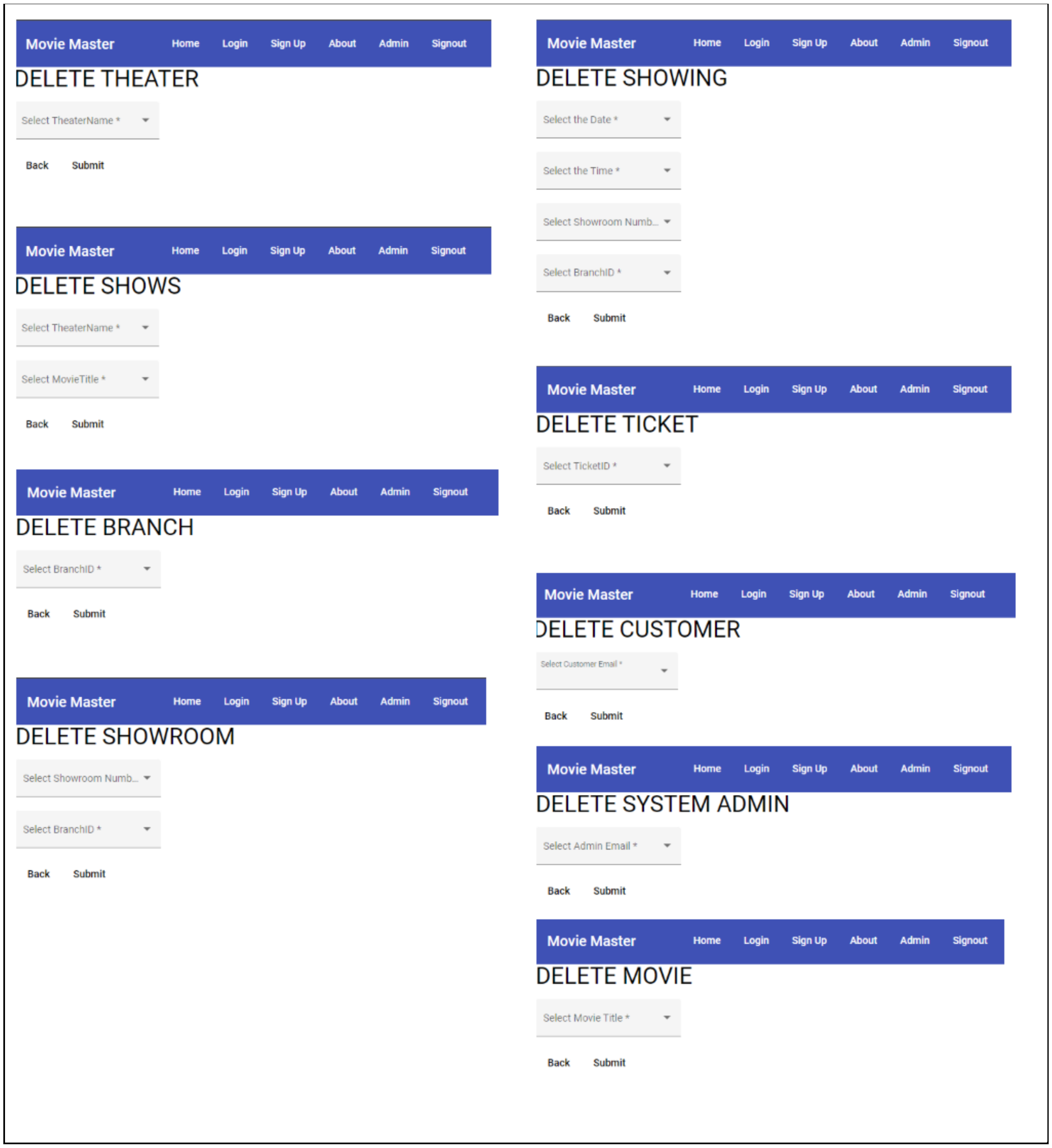


Figure 6.3.4: Admin Delete Pages

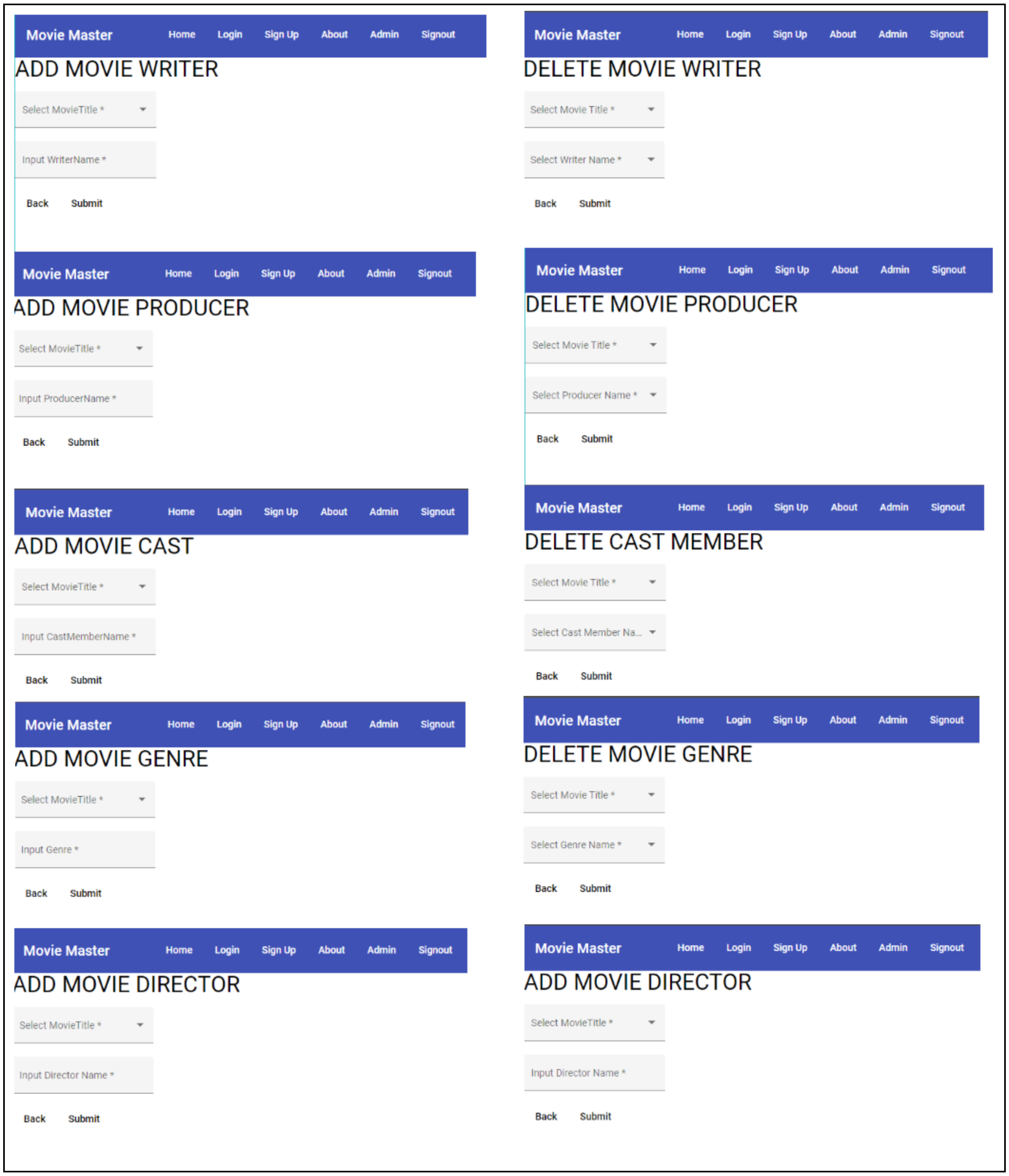


Figure 6.3.5: Admin Add/Delete Movie Attribute Pages