Final Report

Alliana Dela Pena , Sanchit Kumar, Thessalonika Magadia

Abstract

With the first public cinema theatre opening in the United States in 1905, cinema theaters have inevitably evolved over the past 100 years. From buying tickets at a booth at the cinema theatre to buying tickets in advance on a website, viewing movies at a cinema theatre has only become easier. However, this efficient model is still littered with a multitude of small inconveniences that may prevent potential customers from booking seats for a movie resulting in lowered revenue. Some examples of these inconveniences are the lack of being able to order food in advance for a booking or even a fast way to immediately make bookings for a movie showing based on available seats. Cinema theatre chains should have an efficient and fast way to manage multiple individual cinema theatres across multiple regions each with their own movie showings. The system created is the solution to achieve a fast and convenient way to allow customers to book seats and food for a movie showing. On top of this, it hopes to achieve an efficient system that allows an easy way to manage chains across multiple regions and their respective operations.

This system is outlined and displayed in the system description, project design, the implementation section, and the API documentation. Lastly, the user guide outlines the functionalities and the ways in which different users can easily use the system.

Introduction

With most cinema theatres not offering pre-ordering of food for a movie showing and not allowing viewers to easily be able to check available seats for a movie showing, it is crucial to implement a system that can allow for such convenience to be offered to customers. At the same time, with a customer being able to pre-order food, there needs to be systems in place that are both accurate and fast which allow for Employees to keep track of food orders assigned to their theatre and deliver them to the right customer. Lastly, the system needs to be able to efficiently add and remove Movie Showings, and Seats for those Movie Showings, to Cinema Theatres.

System Description

The system created is one that allows for pre-ordering of food easily as well as allowing an easy way to cancel food or tickets ahead of the scheduled movie showing. At the same time, it allows an easy way for customers to determine their expenses for a booking or even check their expenses across multiple bookings. The system also allows for a chain to be managed by a manager. Managers are given certain privileges which includes adding and removing movies as well as editing movie showings. In addition, the system allows employees to complete food orders in an organized manner.

Project Design

This system has 3 users with different roles and, subsequently, different transactions that these users are able to perform.

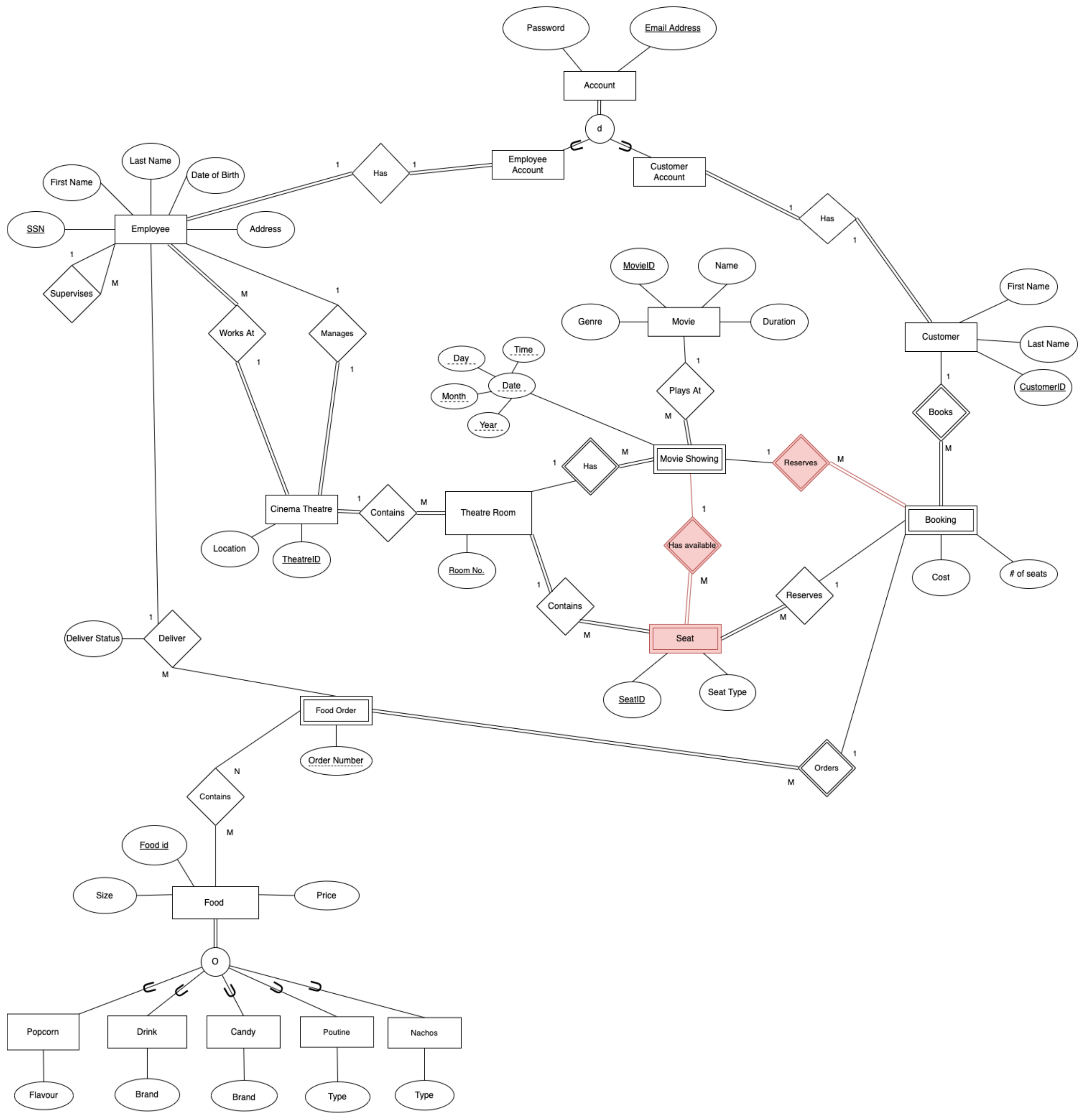
1. Customer: As the cinema theatre system deals with money transactions, it is crucial that customers have to register or login to be able to guarantee that these money transactions and orders can be tracked to an account. Therefore, a customer should be able to create a new account or login to an existing account. Since this is a cinema theatre system, a customer should be able to reserve seats for movie showings of certain movies that they choose. This would require for there to be movie showings for movies in the database and also for there to be seats available for this specific movie showing. A customer should not be able to book seats for a movie showing if the movie showing doesn't exist or there are no seats available for the movie showing. Another big functionality of the customer user is the option to be able to view all of the food available and buy food for movie showings for which they have booked seats for. A customer should not be able to buy food for a movie showing if they have not booked any seats for movie showings. Lastly, it is important that a customer can view their account information as well as any bookings they have created and any food that they have ordered for that booking. Bookings and food orders for those bookings should only be visible to a customer if that customer has actually booked a movie showing and ordered food for that movie showing. If the customer wishes, they should be allowed to cancel all the food that they have ordered for a movie showing prior to the movie showing itself. They should also be able to cancel an entire booking for a movie showing, which should subsequently cancel all food ordered for that movie showing, prior to the movie showing itself.

2. Employee: To ensure that only employees get access to food orders associated with the theatre they work in, a login system is important to identify which food orders should be displayed. The login info is used to identify which theatre the employee works in and is then used to retrieve food orders that are associated with bookings for that theatre. Each food order displays information such as delivery status, order number, date and time, room number, seat ids and the food items and their quantity needed for an order. The list of food orders are listed in a way that the orders that are close to the movie start time are closer to the top. Moreover, orders that have a delivery status 'In progress' are placed higher in their specific time which lets employees know to work on another order. In addition, employees have the ability to modify the delivery status to let their coworkers know if they delivered the order ,are working on an order, or haven't gotten to it yet. When an employee updates the delivery status to 'In progress' or 'Delivered' their SSN is recorded for that order in the database. Also, once an employee updates the delivery status to 'Delivered', the order does not show up on the employee food order page but is still stored in the database until the manager removes it. Lastly, an employee is able to view their account information which includes, first name, last name, birthdate, address, email and theatre id.

3. Manager: Managers are responsible for a very important part of cinema day-to-day functionality, they are responsible for adding movies and movie showings. After logging into the system, managers are able to see a summary of their account information which includes: first name, last name, email, birthdate, address and theatre location. On the

next page, managers can add movies to the database by inputting: movie name, genre and duration. On this same page managers can also view existing movies in the database and remove them from the database. Lastly, managers are able to edit movie showings for the theatre they manage (this is determined by the theatre ID they have linked to their account). Managers can add a movie showing by selecting a movie (that exists in the database), room number (of a room in their theatre), date, and start time. On this same page managers can view existing showings and remove showings for their theatre.

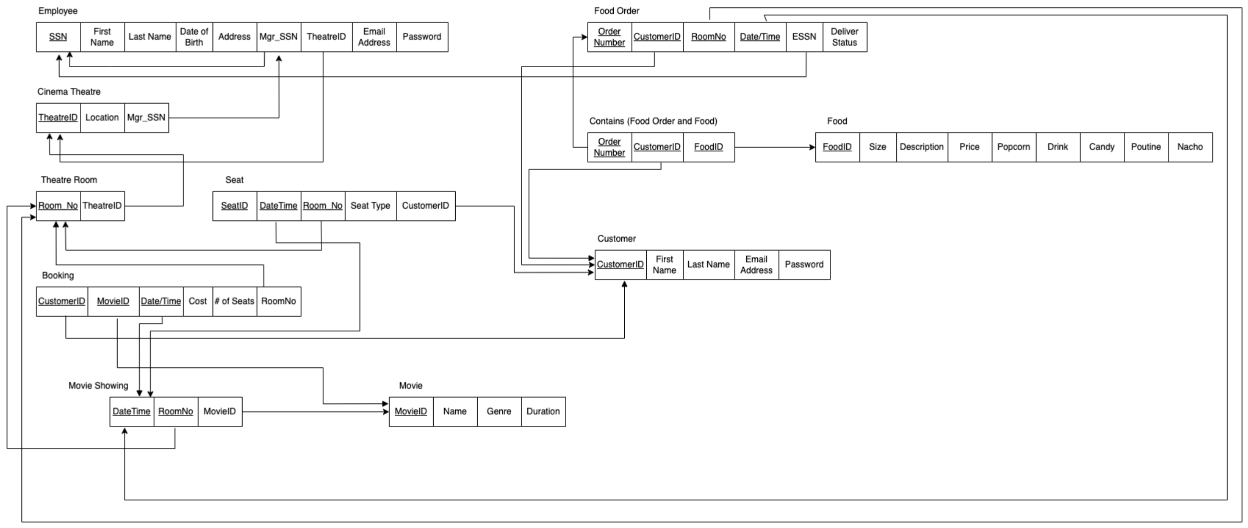
ER Diagram



The changes we made for the ER diagram were mainly to differentiate between the different food orders and seats. We decided to make Booking have an identifying relationship with Movie Showing which demonstrates how a booking does not exist without a Movie Showing. This change caused Food Order to have the Date and RoomNo attribute. We also made Seat a weak entity with an identifying relationship with Movie Showing. This change allowed Seat to have the Date attribute. The supervises relationship is used to connect an employee to their manager in a hierarchical structure. However, we noticed that currently, it does not have a use in the sense that there is nothing done with this relationship. It is still in the ER diagram because this relationship can be used to implement other features that may be useful. For example, it can be used as a way for a manager to monitor employees that work at the Cinema Theatre that they manage. The reasons for these changes are further explained in the Relational Model Diagram section below.

Implementation Section

Relational Model Diagram



There were a number of different changes made to the relational model. First was the addition of DateTime in the Seat table which references the DateTime in Movie Showing. This change was made after much deliberation as we believed that the system would be better off with having temporary seats that were tied to Movie Showings that the Manager user added. This would allow for the database to hold multiple Movie Showings in the same Room of a Cinema Theatre but just at different times. As a result, Customer users would be able to book for Movie Showings that occur at different times, but occur in the same Room of a Cinema Theatre, because a Seat in a Room of a Cinema Theatre would be tied to the date and time during which the Movie Showing is occurring. Another change made to the relational model is the inclusion of DateTime in Food Order that references the DateTime in Movie Showing. The reason for this inclusion is similar to the inclusion of DateTime in Seat. It allows for a Customer user to have multiple food orders, in the database, that occur on different dates based on the Movie Showing that the user has booked seats for.

The DBMS used to implement this system was MySQL. The database created in MySQL was queried using PHP (PDO was used to query the database for the API endpoints and MySQLi was used to query the database for the website).

SQL Queries used:

addCustomerAccount(): INSERT INTO Customer (FirstName, LastName, Email, Password) VALUES (cfirstname, clastname, cemail, cpassword);

addEmployeeAccount(): INSERT INTO Employee (SSN, First_Name, Last_Name, DOB, Address, Mgr_SSN, TheatreID, Email_Address, Password) VALUES
(essn, fname, lname, dob, address, mssn, theatreid, eemail, epassword);

addMovie(): INSERT INTO Movie(Name, Genre, Duration) VALUES (mname, mgenre, mduration);

addMovieShowing(): INSERT INTO movie_showing VALUES (dt, rm, mid);

addSeatsForShowing(): INSERT INTO seat (SeatID, Seat_Type, Room_No, CustomerID, DateTime) VALUES
('A01', 'Wheelchair', roomNo, NULL, dt),
('A02', 'Wheelchair', roomNo, NULL, dt),
('A03', 'Normal', roomNo, NULL, dt),
('A04', 'Normal', roomNo, NULL, dt),
('A05', 'Normal', roomNo, NULL, dt),
('B01', 'Wheelchair', roomNo, NULL, dt),
('B02', 'Wheelchair', roomNo, NULL, dt),
('B03', 'Normal', roomNo, NULL, dt),
('B04', 'Normal', roomNo, NULL, dt),
('B05', 'Normal', roomNo, NULL, dt),
('C01', 'Normal', roomNo, NULL, dt),
('C02', 'Normal', roomNo, NULL, dt),
('C03', 'Normal', roomNo, NULL, dt),
('C04', 'Normal', roomNo, NULL, dt),
('C05', 'Normal', roomNo, NULL, dt),
('D01', 'Normal', roomNo, NULL, dt),
('D02', 'Normal', roomNo, NULL, dt),
('D03', 'Normal', roomNo, NULL, dt),
('D04', 'Normal', roomNo, NULL, dt),
('D05', 'Normal', roomNo, NULL, dt),
('E01', 'Normal', roomNo, NULL, dt),
('E02', 'Normal', roomNo, NULL, dt),
('E03', 'Normal', roomNo, NULL, dt),
('E04', 'Normal', roomNo, NULL, dt),

('E05', 'Normal', roomNo, NULL, dt),
('F01', 'Normal', roomNo, NULL, dt),
('F02', 'Normal', roomNo, NULL, dt),
('F03', 'Normal', roomNo, NULL, dt),
('F04', 'Normal', roomNo, NULL, dt),
('F05', 'Normal', roomNo, NULL, dt);

cancelBookingSeats(): UPDATE Seat SET seat.CustomerID = NULL WHERE seat.CustomerID = customerID AND seat.Room_No = roomNo AND seat.DateTime = dateTime;

checkCustomerAccount(): SELECT CustomerID FROM Customer WHERE Customer.Email = cemail AND Customer.Password = cpassword;

checkManagerSSN(): SELECT Mgr_SSN FROM Cinema_Theatre WHERE Mgr_SSN = mssn;

createBooking(): INSERT INTO Booking (booking.CustomerID, booking.MovieID, booking.DateTime, booking.Cost, booking.No_of_seats, booking.Room_No) VALUES (customerID, movieID, dateTime, cost, noOfSeats, roomNo);

createContainsFoodOrder(): INSERT INTO contains_food_order (Order_Number, CustomerID, FoodID, Quantity) VALUES (orderNumber, customerID, foodID, quantity);

createFoodOrder(): INSERT INTO food_order (CustomerID, RoomNo, DateTime, ESSN, Deliver_Status) VALUES (customerID, roomNo, dateTime, NULL, false);

deleteBooking(): DELETE FROM Booking WHERE booking.CustomerID = customerID AND booking.MovieID = movieID AND booking.DateTime = dateTime AND booking.Room_No = roomNo;

deleteCustomerFood(): DELETE FROM contains_food_order WHERE contains_food_order.Order_Number = orderNumber AND contains_food_order.CustomerID = customerID;

deleteCustomerFoodOrder(): DELETE FROM food_order WHERE food_order.Order_Number = orderNumber AND food_order.CustomerID = customerID AND food_order.RoomNo = roomNo AND food_order.DateTime = dateTime;

getAllFood(): SELECT * FROM food;

getAllFoodCost(): SELECT contains_food_order.Quantity, Food.Price FROM contains_food_order, Food WHERE contains_food_order.Order_Number = orderNumber AND contains_food_order.CustomerID = customerID AND contains_food_order.FoodID = Food.FoodID;

getAllMovie(): SELECT * FROM movie;

getAllShowings(): SELECT m.Name "Name", ms.RoomNo "RoomNo", ms.DateTime "DateTime"
FROM movie_showing as ms, movie as m, theatre_room as t
WHERE t.Room_No = ms.RoomNo and m.MovieID = ms.MovieID and t.TheatreID = id
ORDER BY DateTime;

getAvailableMovieShowings(): SELECT Cinema_Theatre.Location, Movie_Showing.RoomNo, Movie_Showing.DateTime, Movie.Name, Movie.Genre, Movie.Duration FROM Movie_Showing, Movie, Theatre_Room, Cinema_Theatre WHERE Movie_Showing.DateTime > dateTime AND Movie_Showing.MovieID = Movie.MovieID AND Movie_Showing.RoomNo = Theatre_Room.Room_No AND Theatre_Room.TheatreID = Cinema_Theatre.TheatreID;

getBookingCost(): SELECT booking.Cost FROM Booking WHERE booking.CustomerID = customerID AND booking.MovieID = movieID AND booking.DateTime = dateTime AND booking.Room_No = roomNo;

getBookingCustomer(): SELECT booking.CustomerID FROM Booking WHERE booking.CustomerID = customerID AND booking.MovieID = movieID AND booking.DateTime = dateTime AND booking.Room_No = roomNo;

getCustomerBookings(): SELECT Movie.Name, Movie.Genre, Movie.Duration, Booking.DateTime, Booking.Cost, Booking.Room_No, Cinema_Theatre.Location, Booking.MovieID FROM Movie, Booking, Cinema_Theatre, Theatre_Room WHERE DATE(Booking.DateTime) > dateTime AND Booking.CustomerID = customerID AND Booking.MovieID = Movie.MovieID AND Booking.Room_No = Theatre_Room.Room_No AND Theatre_Room.TheatreID = Cinema_Theatre.TheatreID;

getCustomerFood(): SELECT contains_food_order.FoodID, contains_food_order.Quantity FROM food_order, contains_food_order WHERE food_order.CustomerID = customerID AND food_order.RoomNo = roomNo AND food_order.DateTime = dateTime AND food_order.Order_Number = contains_food_order.Order_Number AND food_order.CustomerID = contains_food_order.CustomerID;

getCustomerInfo(): SELECT * FROM customer WHERE customer.Email = cemail AND customer.Password = cpassword;

getCustomerSeat(): SELECT Seat.SeatID, Seat.Seat_Type FROM Seat WHERE Seat.Room_No = roomNo AND Seat.CustomerID = customerID AND Seat.DateTime = dateTime;

getEmployeeInfo(): SELECT * FROM Employee WHERE Employee.Email_Address = eemail AND Employee.Password = epassword;

getFoodByID(): SELECT * FROM Food WHERE FoodID = food;

getFoodItemsbyOrderNo(): SELECT * FROM Contains_Food_Order, Food WHERE Contains_Food_Order.Order_Number = orderNo AND Contains_Food_Order.FoodID = Food.FoodID;

getFoodOrders(): SELECT DISTINCT Food_Order.Order_Number, Food_Order.DateTime, Seat.Room_No, Food_Order.CustomerID, Food_Order.Deliver_Status, Food_Order.ESSN
From Food_Order, Seat, Theatre_Room, Employee
WHERE Deliver_Status <> delivered AND Food_Order.DateTime > today AND Seat.DateTime = Food_Order.DateTime AND Seat.CustomerID = Food_Order.CustomerID AND Seat.Room_No = Theatre_Room.Room_No AND Theatre_Room.TheatreID = Employee.TheatreID AND Employee.Email_Address = eemail
ORDER BY Food_Order.Deliver_Status DESC, Food_Order.DateTime;

getFoodPrice(): SELECT Food.Price FROM Food WHERE Food.FoodID = food;

getFoodQuantity(): SELECT contains_food_order.Quantity FROM contains_food_order WHERE contains_food_order.Order_Number = orderNumber AND contains_food_order.CustomerID = customerID AND contains_food_order.FoodID = foodID;

getManagerAndTheatreInfo(): SELECT Mgr_SSN, TheatreID FROM Cinema_Theatre WHERE Location = loc;

getMovieID(): SELECT movie_showing.MovieID FROM movie_showing WHERE movie_showing.DateTime = dateTime AND movie_showing.RoomNo = roomNo;

getMovieLength(): SELECT Duration FROM movie WHERE MovieID = id;

getMovieShowingFood(): SELECT Movie.Name, Booking.DateTime, Booking.Room_No FROM Movie, Booking WHERE DATE(Booking.DateTime) > dateTime AND Booking.CustomerID = customerID AND Booking.MovieID = Movie.MovieID;

getMovieShowingSeats(): SELECT Seat.SeatID, Seat.Seat_Type FROM Seat, Movie_Showing WHERE Movie_Showing.RoomNo = roomNo AND Seat.DateTime = dateTime AND Movie_Showing.RoomNo = Seat.Room_No AND Seat.CustomerID IS NULL AND Movie_Showing.DateTime = dateTime;

getOrderNumber(): SELECT food_order.Order_Number FROM food_order WHERE food_order.CustomerID = customerID AND food_order.RoomNo = roomNo AND food_order.DateTime = dateTime;

getSeatIDs(): SELECT Seat.SeatID FROM Seat WHERE Seat.CustomerID = customerid AND Seat.DateTime = dt;

getShowingsInRoom(): SELECT m.Name "Name", m.Duration "Duration", ms.RoomNo "RoomNo", ms.DateTime "DateTime"
FROM movie_showing as ms, movie as m, theatre_room as t
WHERE t.Room_No = ms.RoomNo and m.MovieID = ms.MovieID and t.Room_No = room
ORDER BY DateTime;

getTheatreLocation(): SELECT Location FROM cinema_theatre WHERE TheatreID = id;

getTheatreRooms(): SELECT Room_No FROM theatre_room WHERE TheatreID= id;

removeMovie(): DELETE FROM movie WHERE MovieID = id;

removeMovieShowing(): DELETE FROM movie_showing WHERE DateTime = date AND RoomNo = room;

updateBookingCost(): UPDATE Booking SET booking.Cost = cost WHERE booking.CustomerID = customerID AND booking.MovieID = movieID AND booking.DateTime = dateTime AND booking.Room_No = roomNo;

updateCostNofSeatsBooking(): UPDATE Booking SET booking.Cost = cost, booking.No_of_seats = noOfSeats WHERE booking.CustomerID = customerID AND booking.MovieID = movieID AND booking.DateTime = dateTime AND booking.Room_No = roomNo;

updateCustomerSeat(): UPDATE Seat SET seat.CustomerID = cID WHERE seat.SeatID = sID AND seat.Room_No = roomNo AND seat.DateTime = dateTime;

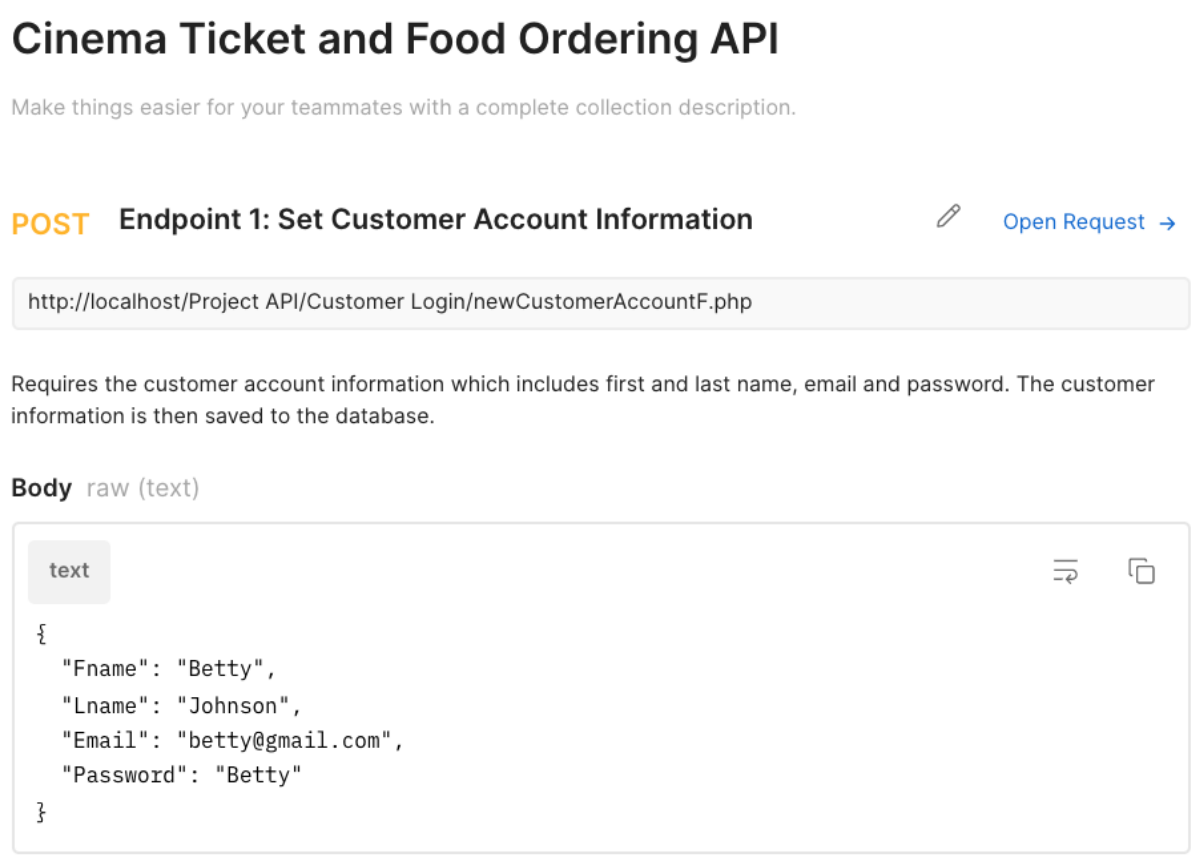
updateDeliverStatus(): UPDATE Food_Order SET ESSN = Essn, Deliver_Status = dstatus WHERE Order_Number = orderNo AND CustomerID = customerID AND RoomNo = roomNo AND Food_Order.DateTime = dt;

updateQuantity(): UPDATE contains_food_order SET contains_food_order.Quantity = quantity WHERE contains_food_order.Order_Number = orderNumber AND contains_food_order.CustomerID = customerID AND contains_food_order.FoodID = foodID;

UPDATE Seat SET CustomerID = ? WHERE SeatID = ? AND Room_No = ? AND DateTime = ?

API Documentation

Link to Postman Documentation:
https://documenter.getpostman.com/view/18775667/UVRAJ7Ry



Cinema Ticket and Food Ordering API

Make things easier for your teammates with a complete collection description.

POST **Endpoint 1: Set Customer Account Information** Open Request →

```
http://localhost/Project API/Customer Login/newCustomerAccountF.php
```

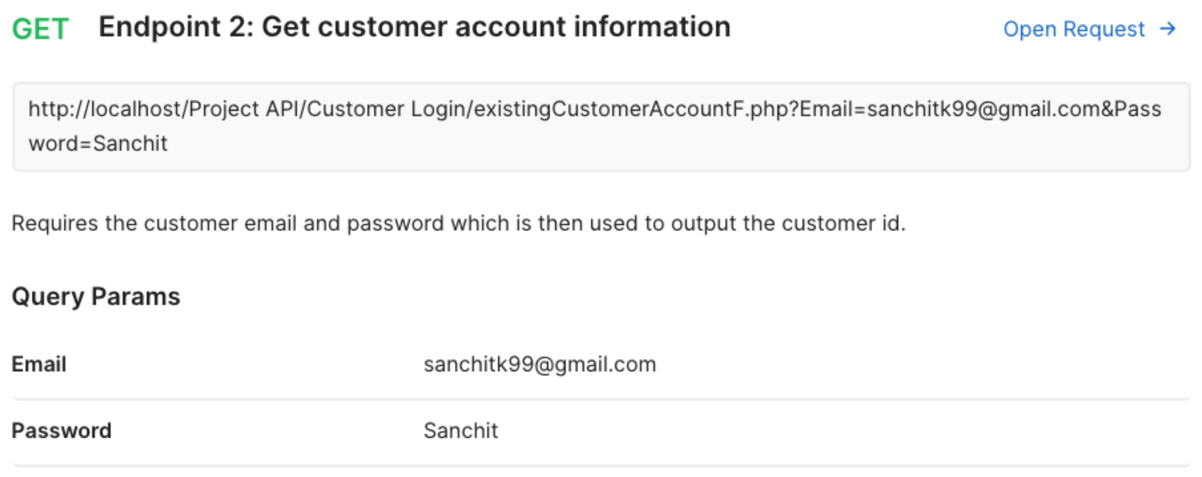
Requires the customer account information which includes first and last name, email and password. The customer information is then saved to the database.

Body raw (text)

text

```
{
  "Fname": "Betty",
  "Lname": "Johnson",
  "Email": "betty@gmail.com",
  "Password": "Betty"
}
```

Response: Customer ID: 13



GET **Endpoint 2: Get customer account information** Open Request →

```
http://localhost/Project API/Customer Login/existingCustomerAccountF.php?Email=sanchitk99@gmail.com&Pass
word=Sanchit
```

Requires the customer email and password which is then used to output the customer id.

Query Params

Email	sanchitk99@gmail.com
Password	Sanchit

Response: Customer ID: 1

GET **Endpoint 2: Customer Account Info Page** Open Request →

```
http://localhost/Project API/Customer Pages/customerAccountInfoF.php?Email=sanchitk99@gmail.com&Password=Sanchit
```

Requires the customer email and password which is then used to output the customer information which includes first and last name and email.

Query Params

Email	sanchitk99@gmail.com
Password	Sanchit

Response: First Name: Sanchit Last Name: Kumar Email: sanchitk99@gmail.com

POST **Endpoint 3: Set Employee Account Information** Open Request →

```
http://localhost/Project API/Employee Login/newEmployeeAccountF.php
```

Requires the employee SSN, first and last name, date of birth, address, email, password and work location. The employee information is then saved to the database.

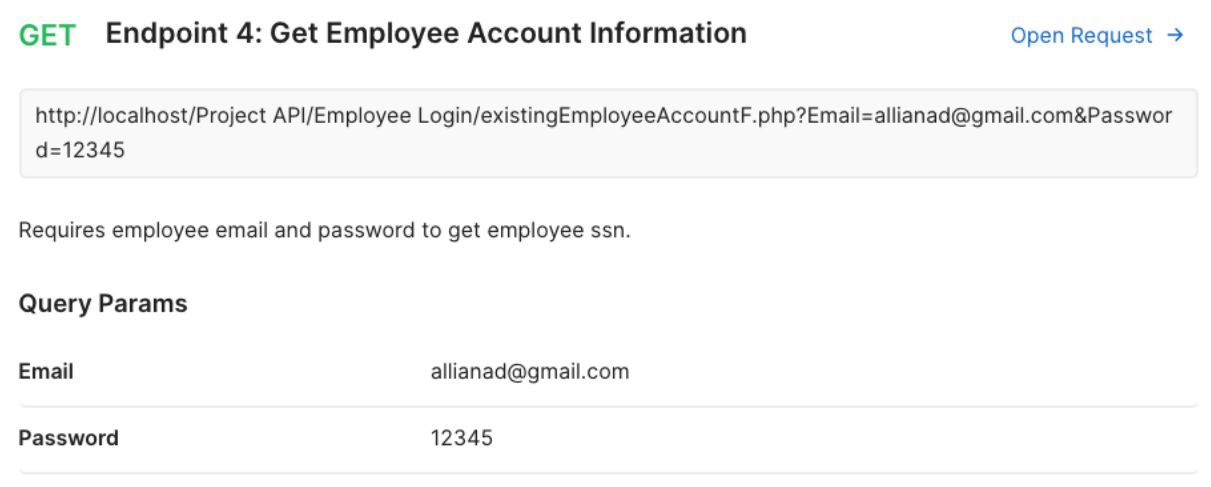
Body raw (text)

text

```
{
  "SSN": "999999999",
  "Fname": "Vicky",
  "Lname": "Lin",
  "DateOfBirth": "1987-04-02",
  "Address": "1934 Butter Blvd",
  "Email": "vickyl@gmail.com",
  "Password": "vicky",
  "WorkLocation": "Calgary"
}
```

View more

Response: Employee SSN: 999999999



GET **Endpoint 4: Get Employee Account Information** Open Request →

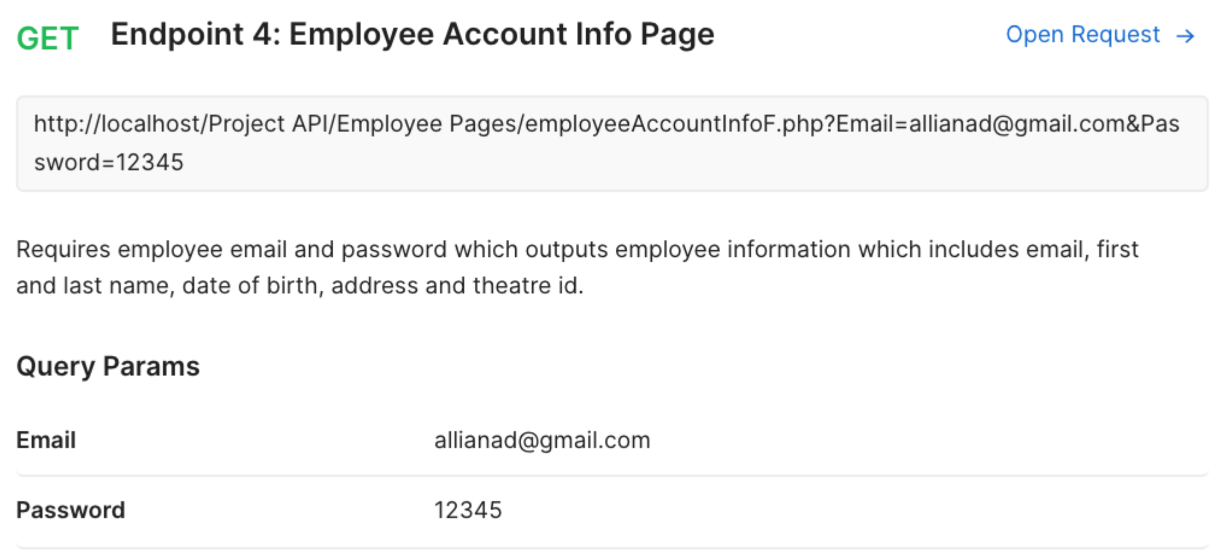
```
http://localhost/Project API/Employee Login/existingEmployeeAccountF.php?Email=allianad@gmail.com&Password=12345
```

Requires employee email and password to get employee ssn.

Query Params

Email	allianad@gmail.com
Password	12345

Response:
SSN: 2
Employee Account



GET **Endpoint 4: Employee Account Info Page** Open Request →

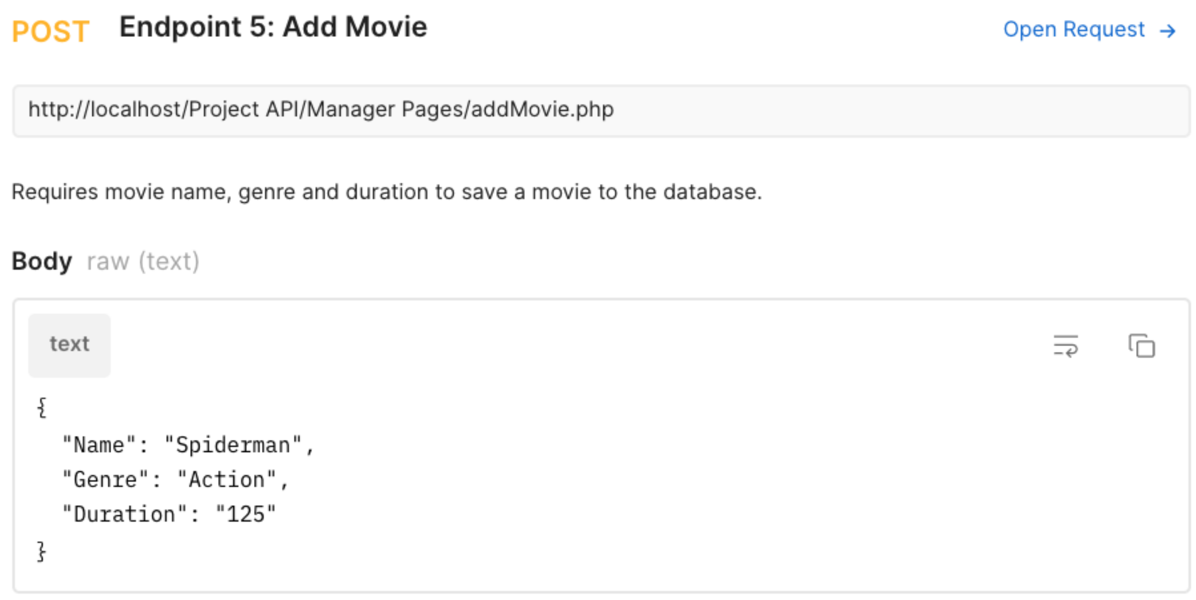
```
http://localhost/Project API/Employee Pages/employeeAccountInfoF.php?Email=allianad@gmail.com&Password=12345
```

Requires employee email and password which outputs employee information which includes email, first and last name, date of birth, address and theatre id.

Query Params

Email	allianad@gmail.com
Password	12345

Response:
Email: allianad@gmail.com
Fname: Alliana
Lname Dela Pena
Date of Birth: 1941-05-24
Address: 52 Rainbow Road
TheatreID: 1


POST **Endpoint 5: Add Movie**

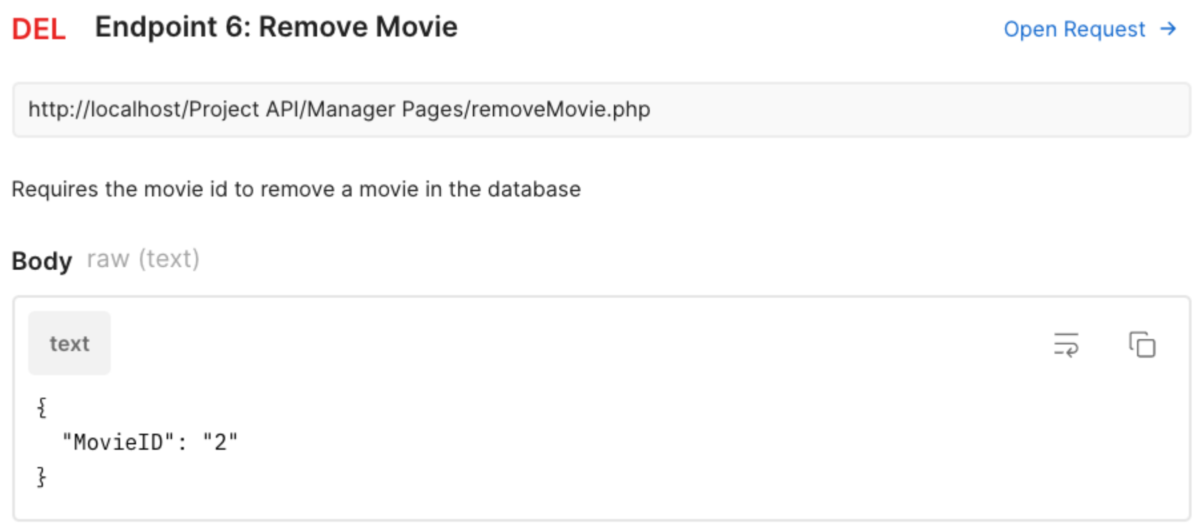
http://localhost/Project API/Manager Pages/addMovie.php

Requires movie name, genre and duration to save a movie to the database.

Body raw (text)

```
{
  "Name": "Spiderman",
  "Genre": "Action",
  "Duration": "125"
}
```

Response:
Movie added to database.


DEL **Endpoint 6: Remove Movie**

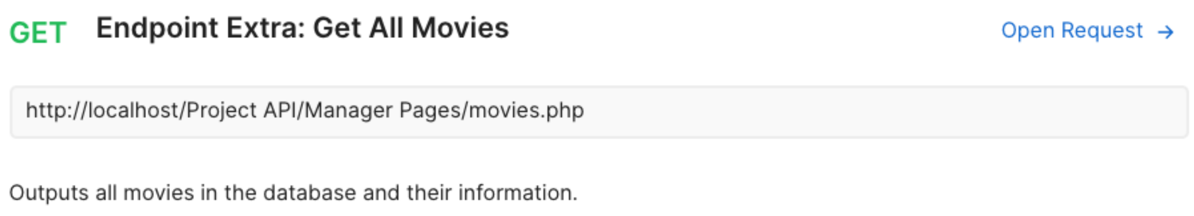
http://localhost/Project API/Manager Pages/removeMovie.php

Requires the movie id to remove a movie in the database

Body raw (text)

```
{
  "MovieID": "2"
}
```

Response:
Movie, with id = 2 removed from database.



GET **Endpoint Extra: Get All Movies** Open Request →

```
http://localhost/Project API/Manager Pages/movies.php
```

Outputs all movies in the database and their information.

Response:

```
{
   "movies": [
      {
         "Name": "Encanto",
         "Genre": "Family",
         "Duration": "109"
      },
      {
         "Name": "Spiderman",
         "Genre": "Action",
         "Duration": "125"
      }
   ]
}
```

GET Endpoint 7 & 10: Get Available Movie Showings

Open Request →

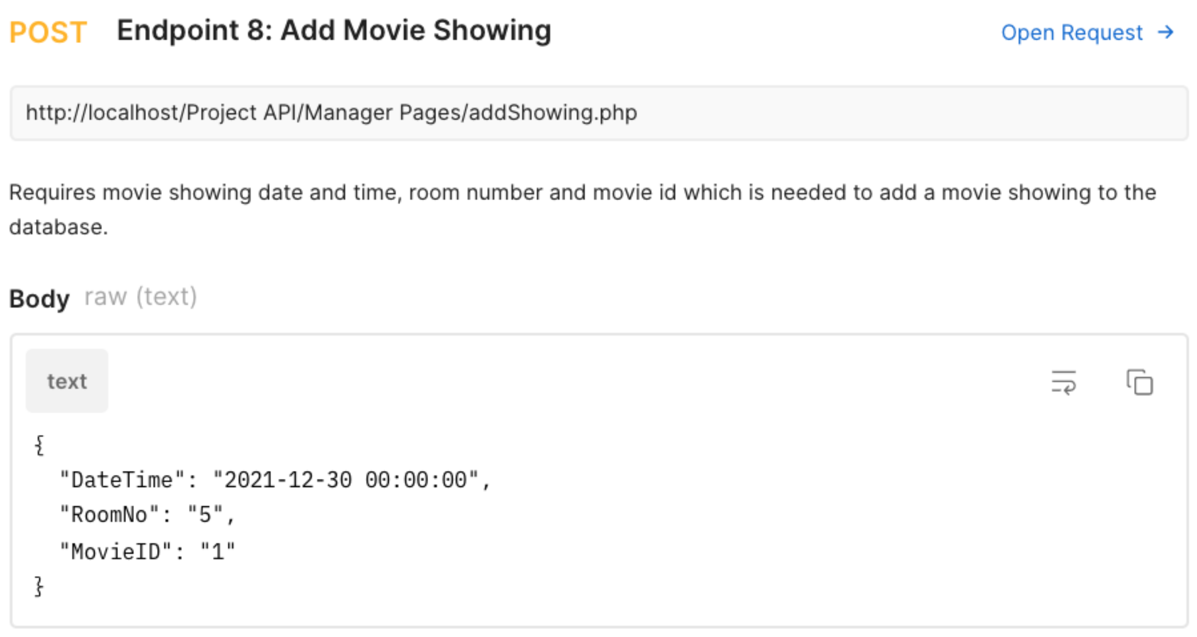
```
http://localhost/Project API/Customer Pages/customerHomeF.php
```

Outputs all available movie showings as well as the available seats for that showing.

Response:

```
{
   "Movie Showing": [
      {
         "Information": [
            {
               "Name": "Encanto",
               "Genre": "Family",
               "Duration": "109",
               "Date & Time": "2021-12-25 17:30:00",
               "Location": "Calgary"
            }
         ],
         "Seats": []
      },
      {
         "Information": [
            {
               "Name": "Encanto",
               "Genre": "Family",
               "Duration": "109",
               "Date & Time": "2021-12-25 13:30:00",
               "Location": "Calgary"
            }
         ],
         "Seats": [
            {
               "Seat ID": "D01",
               "Seat Type": "Wheelchair"
            },
            {
               "Seat ID": "D02",
               "Seat Type": "Normal"
            }
         ]
      }
   ]
```

}



POST **Endpoint 8: Add Movie Showing** Open Request →

http://localhost/Project API/Manager Pages/addShowing.php

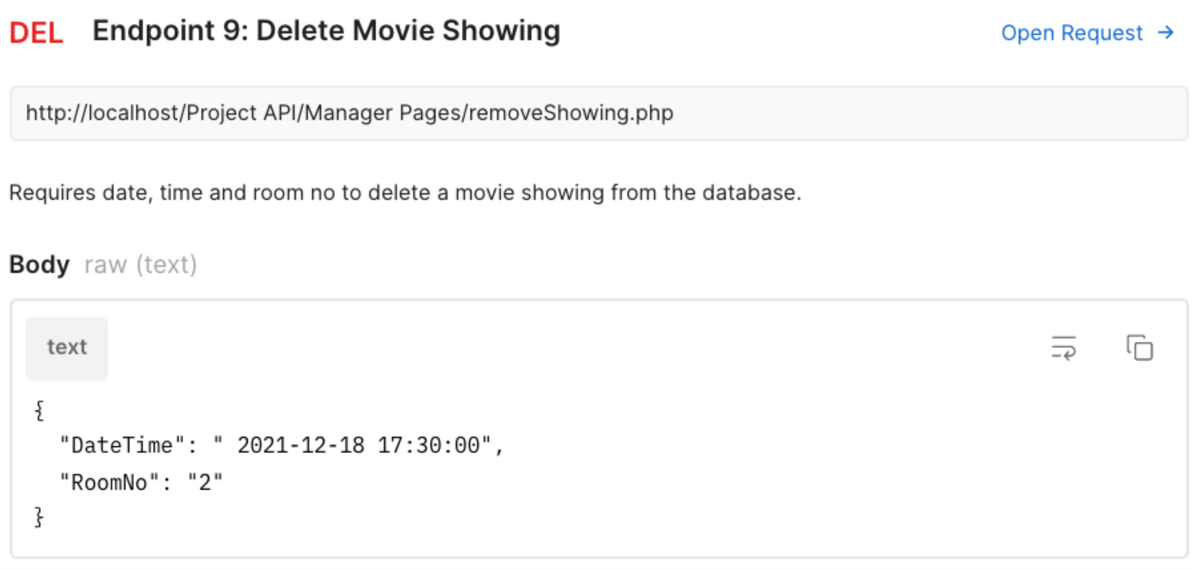
Requires movie showing date and time, room number and movie id which is needed to add a movie showing to the database.

Body raw (text)

text

```
{
  "DateTime": "2021-12-30 00:00:00",
  "RoomNo": "5",
  "MovieID": "1"
}
```

Response:
Showing for movieID = 1 in room 5 on 2021-12-30 00: 00: 00 has been added to the database, and corresponding seats have been added



DEL **Endpoint 9: Delete Movie Showing** Open Request →

http://localhost/Project API/Manager Pages/removeShowing.php

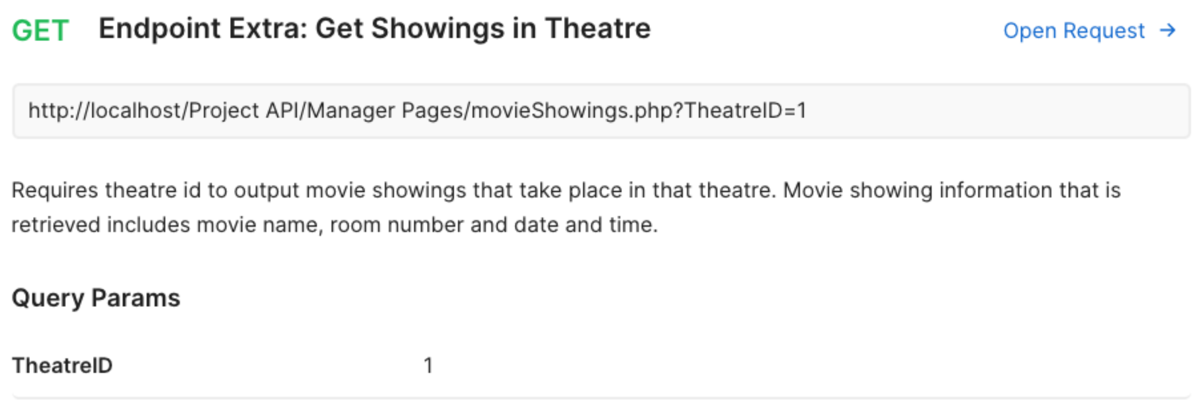
Requires date, time and room no to delete a movie showing from the database.

Body raw (text)

text

```
{
  "DateTime": " 2021-12-18 17:30:00",
  "RoomNo": "2"
}
```

Response:
Movie Showing in room = 2 on 2021-12-18 17: 30: 00 removed from database.



GET **Endpoint Extra: Get Showings in Theatre** Open Request →

```
http://localhost/Project API/Manager Pages/movieShowings.php?TheatreID=1
```

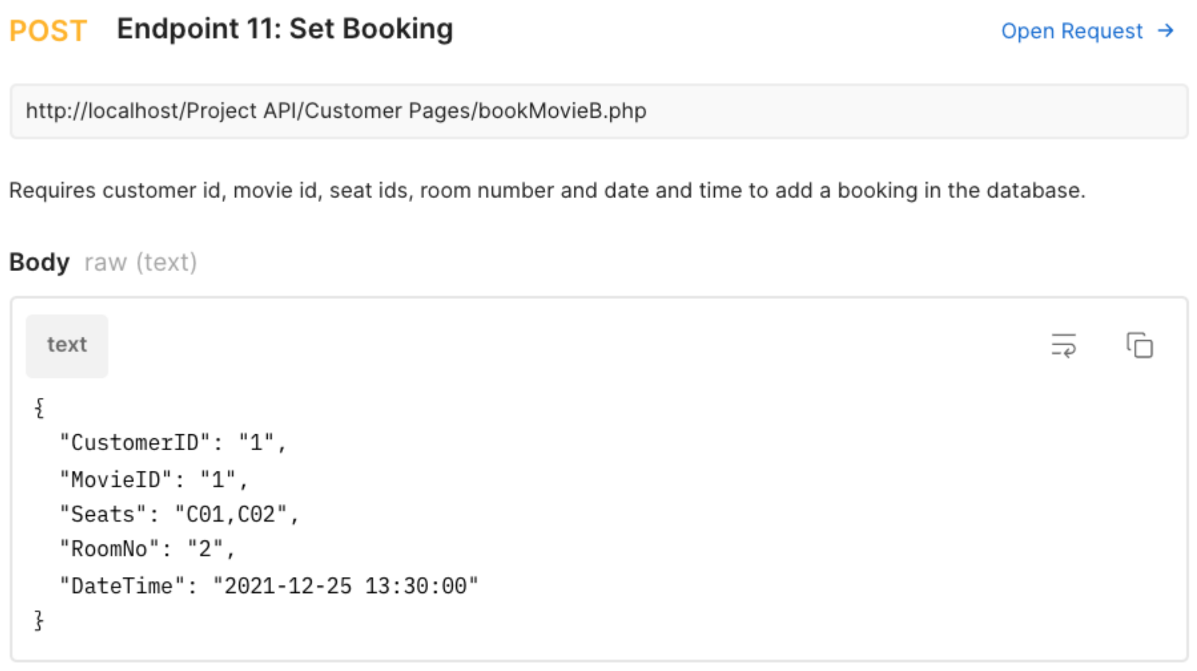
Requires theatre id to output movie showings that take place in that theatre. Movie showing information that is retrieved includes movie name, room number and date and time.

Query Params

TheatreID	1

Response:

```
{
   "movie_showings": [
      {
         "Name": "Encanto",
         "RoomNo": "2",
         "DateTime": "2021-12-18 13:00:00"
      },
      {
         "Name": "Encanto",
         "RoomNo": "2",
         "DateTime": "2021-12-25 13:30:00"
      },
      {
         "Name": "Encanto",
         "RoomNo": "2",
         "DateTime": "2021-12-25 17:30:00"
      },
      {
         "Name": "Encanto",
         "RoomNo": "5",
         "DateTime": "2021-12-30 00:00:00"
      }
   ]
}
```



POST **Endpoint 11: Set Booking** Open Request →

http://localhost/Project API/Customer Pages/bookMovieB.php

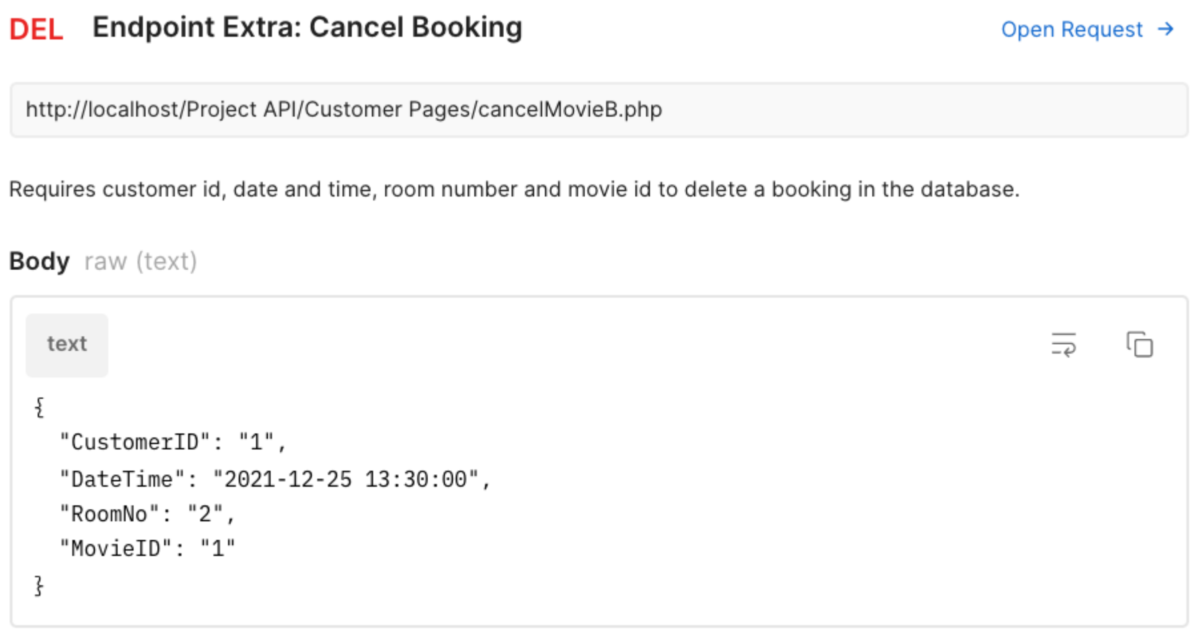
Requires customer id, movie id, seat ids, room number and date and time to add a booking in the database.

Body raw (text)

text

```
{
  "CustomerID": "1",
  "MovieID": "1",
  "Seats": "C01,C02",
  "RoomNo": "2",
  "DateTime": "2021-12-25 13:30:00"
}
```

Response:
Booked: C01C02



DEL **Endpoint Extra: Cancel Booking** Open Request →

http://localhost/Project API/Customer Pages/cancelMovieB.php

Requires customer id, date and time, room number and movie id to delete a booking in the database.

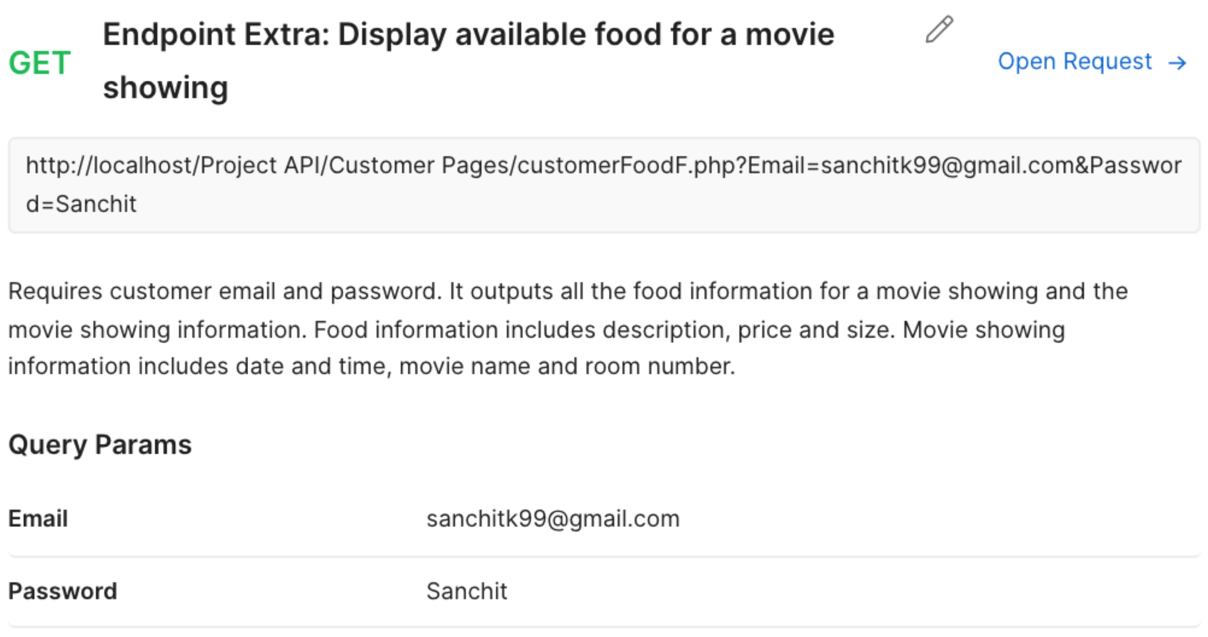
Body raw (text)

text

```
{
  "CustomerID": "1",
  "DateTime": "2021-12-25 13:30:00",
  "RoomNo": "2",
  "MovieID": "1"
}
```

Response:

Booking Deleted Successfully.



GET **Endpoint Extra: Display available food for a movie showing**

Open Request →

```
http://localhost/Project API/Customer Pages/customerFoodF.php?Email=sanchitk99@gmail.com&Password=Sanchit
```

Requires customer email and password. It outputs all the food information for a movie showing and the movie showing information. Food information includes description, price and size. Movie showing information includes date and time, movie name and room number.

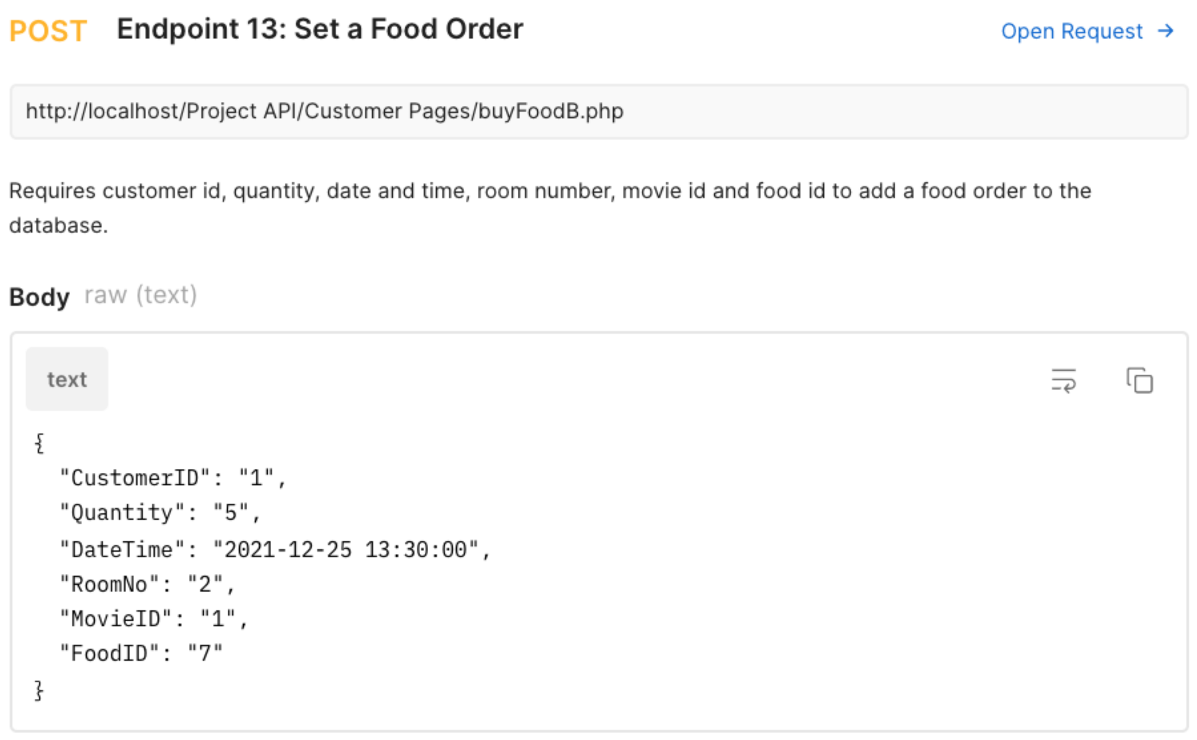
Query Params

Email	sanchitk99@gmail.com
Password	Sanchit

Response:

```
{
  "Movie Showing": [
    {
      "Information": [
        {
          "Date & Time": "2021-12-25 13:30:00",
          "Name": "Encanto",
          "Room No": "2"
        }
      ],
      "Food": [
        {
          "Description": "Coke",
          "Size": "S",
          "Price": "2"
        },
        {
          "Description": "Coke",
          "Size": "M",
          "Price": "4"
        },
        {
          "Description": "Coke",
          "Size": "L",
    "Price": "5"
},
{
    "Description": "Sprite",
    "Size": "S",
    "Price": "2"
},
{
    "Description": "Sprite",
    "Size": "M",
    "Price": "4"
},
{
    "Description": "Sprite",
    "Size": "L",
    "Price": "5"
},
{
    "Description": "Regular",
    "Size": "S",
    "Price": "6"
},
{
    "Description": "Regular",
    "Size": "M",
    "Price": "8"
},
{
    "Description": "Regular",
    "Size": "L",
    "Price": "9"
},
{
    "Description": "Buttered",
    "Size": "S",
    "Price": "8"
},
{
    "Description": "Buttered",
    "Size": "M",
    "Price": "10"
},
{
    "Description": "Buttered",
    "Size": "L",
    "Price": "11"
},
{
    "Description": "Smarties",
    "Size": "R",
    "Price": "4"
},
{
    "Description": "Kit-Kat",
    "Size": "R",
    "Price": "4"
},
{
    "Description": "M&Ms",
    "Size": "R",
    "Price": "4"
},
{
    "Description": "Classic",
    "Size": "R",
    "Price": "8"
},
{
    "Description": "Classic",
    "Size": "L",
    "Price": "11"
},
{
    "Description": "Veggie",
    "Size": "R",
    "Price": "8"
},
{
    "Description": "Veggie",
    "Size": "L",
    "Price": "11"
},
{
    "Description": "Traditional",
    "Size": "R",
    "Price": "9"
},
{
```

```
                "Description": "Traditional",
                "Size": "L",
                "Price": "12"
            },
            {
                "Description": "Beef",
                "Size": "R",
                "Price": "10"
            },
            {
                "Description": "Beef",
                "Size": "L",
                "Price": "13"
            }
        ]
    }
  ]
}
```



POST **Endpoint 13: Set a Food Order** Open Request →

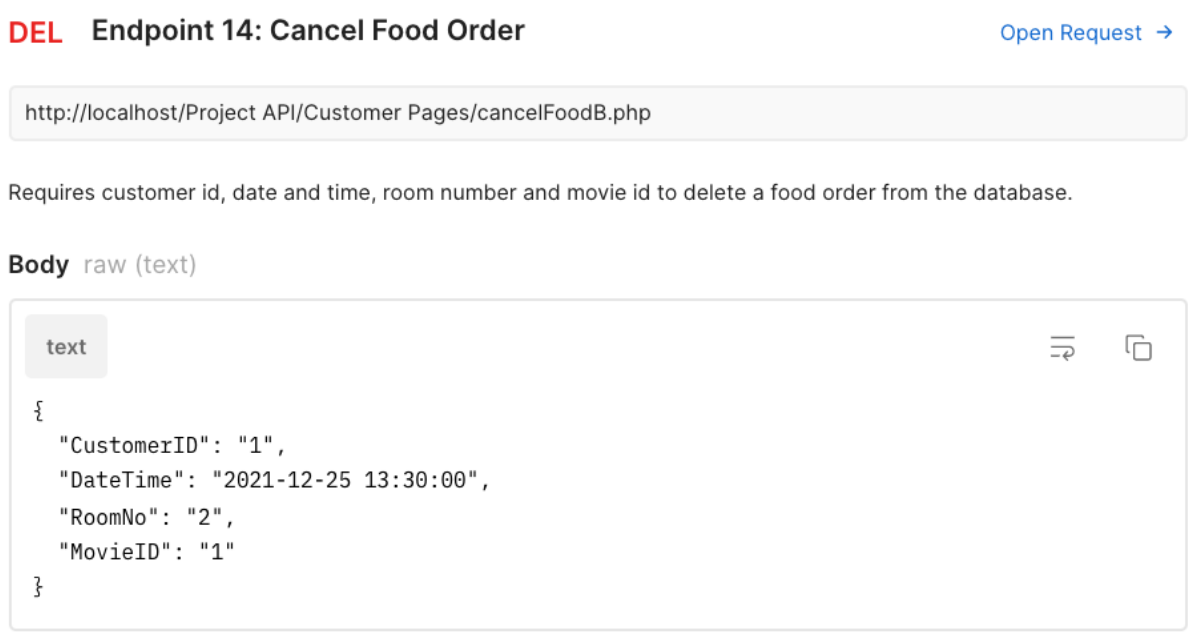
http://localhost/Project API/Customer Pages/buyFoodB.php

Requires customer id, quantity, date and time, room number, movie id and food id to add a food order to the database.

Body raw (text)

```text
{
  "CustomerID": "1",
  "Quantity": "5",
  "DateTime": "2021-12-25 13:30:00",
  "RoomNo": "2",
  "MovieID": "1",
  "FoodID": "7"
}
```

Response:
Order Number: 1 Price: $30



DEL **Endpoint 14: Cancel Food Order**

Open Request →

```
http://localhost/Project API/Customer Pages/cancelFoodB.php
```

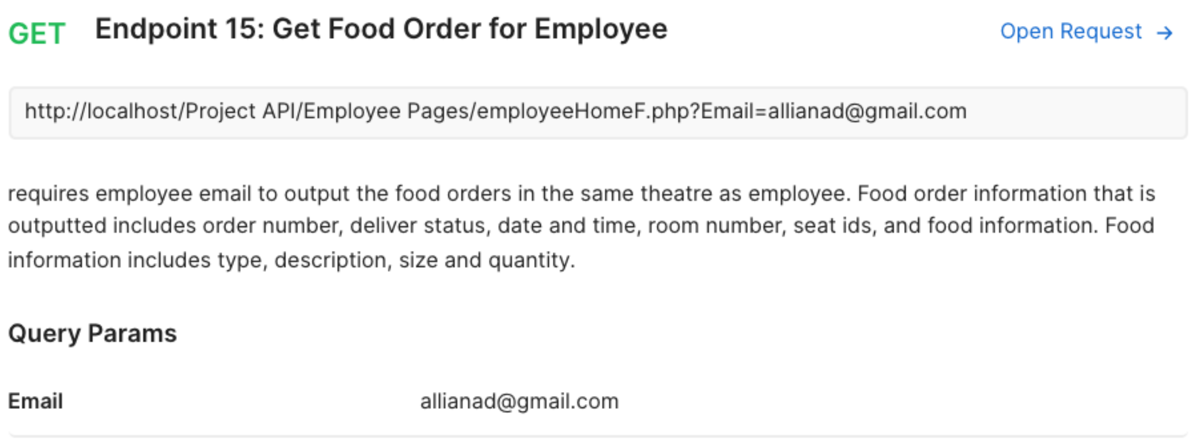
Requires customer id, date and time, room number and movie id to delete a food order from the database.

Body raw (text)

text

```
{
  "CustomerID": "1",
  "DateTime": "2021-12-25 13:30:00",
  "RoomNo": "2",
  "MovieID": "1"
}
```

Response:
Food Order Deleted Successfully.



GET **Endpoint 15: Get Food Order for Employee**

Open Request →

```
http://localhost/Project API/Employee Pages/employeeHomeF.php?Email=allianad@gmail.com
```

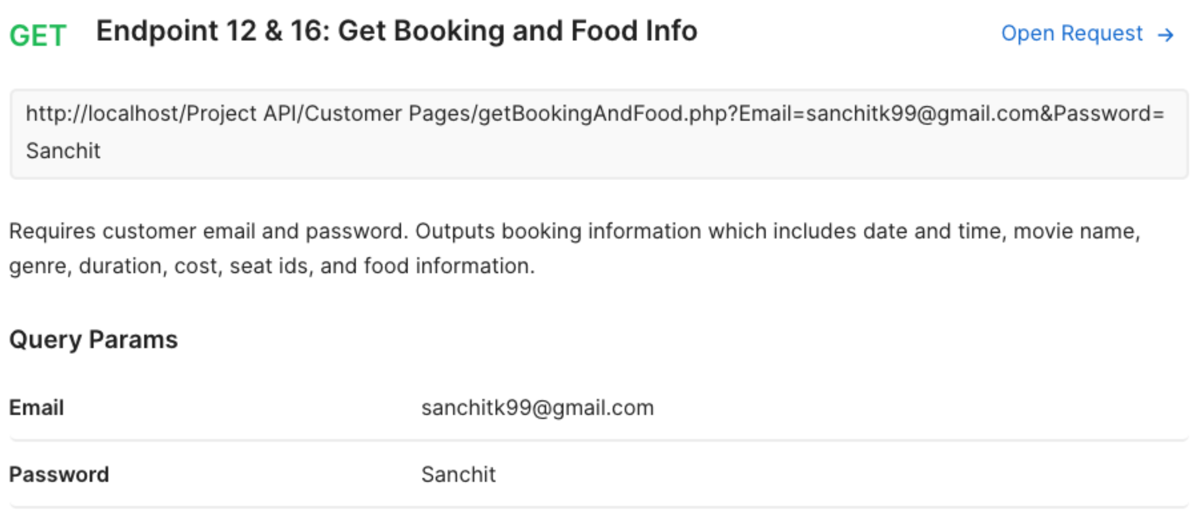
requires employee email to output the food orders in the same theatre as employee. Food order information that is outputted includes order number, deliver status, date and time, room number, seat ids, and food information. Food information includes type, description, size and quantity.

Query Params

Email	allianad@gmail.com

Response:

```
{
  "Food Orders": [
    {
      "Order Number": "2",
      "Deliver Status": "Order Placed",
      "Date Time": "2021-12-25 13:30:00",
      "Room Number": "2",
      "Seat": [
        {
            "Seat ID": "B01"
        },
        {
            "Seat ID": "B02"
        }
    ],
    "Food": [
        {
            "Food": "Drink",
            "Description": "Coke",
            "Size": "S",
            "Quantity": "1"
        },
        {
            "Food": "Nacho",
            "Description": "Beef",
            "Size": "R",
            "Quantity": "2"
        }
    ]
},
{
    "Order Number": "3",
    "Deliver Status": "Order Placed",
    "Date Time": "2021-12-25 13:30:00",
    "Room Number": "2",
    "Seat": [
        {
            "Seat ID": "C01"
        },
        {
            "Seat ID": "C02"
        }
    ],
    "Food": [
        {
            "Food": "Drink",
            "Description": "Coke",
            "Size": "L",
            "Quantity": "2"
        },
        {
            "Food": "Poutine",
          "Description": "Veggie",
          "Size": "L",
          "Quantity": "1"
        }
      ]
    },
    {
      "Order Number": "5",
      "Deliver Status": "Order Placed",
      "Date Time": "2021-12-25 17:30:00",
      "Room Number": "2",
      "Seat": [
        {
          "Seat ID": "E03"
        }
      ],
      "Food": [
        {
          "Food": "Candy",
          "Description": "M&Ms",
          "Size": "R",
          "Quantity": "1"
        }
      ]
    }
  ]
}
```



GET **Endpoint 12 & 16: Get Booking and Food Info** Open Request →

```
http://localhost/Project API/Customer Pages/getBookingAndFood.php?Email=sanchitk99@gmail.com&Password=Sanchit
```

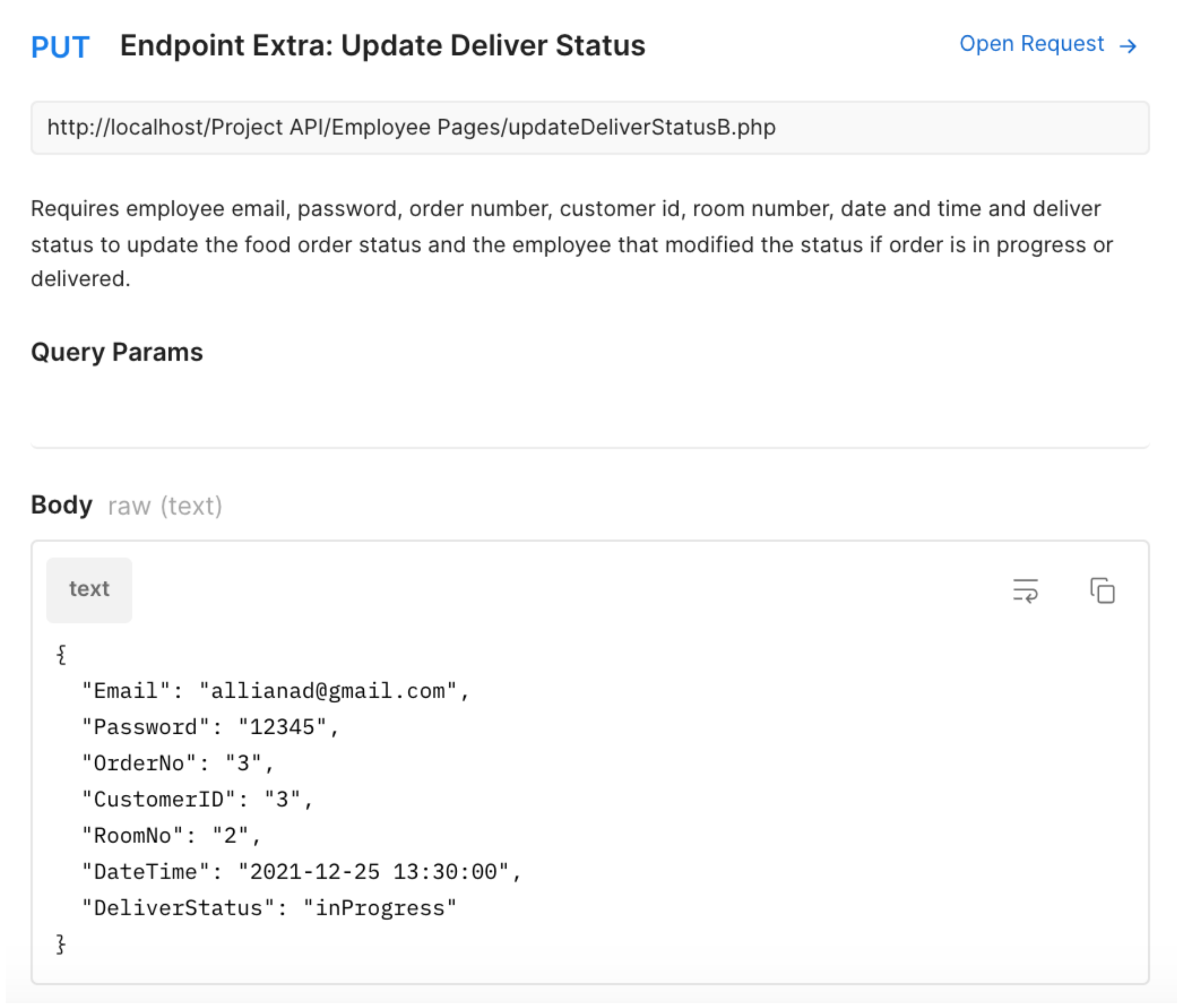
Requires customer email and password. Outputs booking information which includes date and time, movie name, genre, duration, cost, seat ids, and food information.

Query Params

Email	sanchitk99@gmail.com
Password	Sanchit

Response:

```
{
  "Booking": {
    "Movie Showings": [
      {
        "Information": [
          {
            "Date & Time": "2021-12-25 13:30:00",
            "Name": "Encanto",
            "Genre": "Family",
            "Duration": "109",
            "Cost": "36"
          }
        ],
        "Seat": [
          {
            "Seat ID": "A01",
            "Seat Type": "Wheelchair"
          },
          {
            "Seat ID": "A02",
            "Seat Type": "Normal"
          }
        ],
        "Food": [
          {
            "Description": "Sprite",
            "Size": "S",
            "Quantity": "1",
            "Price": "2"
          },
          {
            "Description": "Buttered",
            "Size": "S",
            "Quantity": "1",
            "Price": "8"
          }
        ]
      }
    ]
  }
}
```



PUT **Endpoint Extra: Update Deliver Status** Open Request →

```
http://localhost/Project API/Employee Pages/updateDeliverStatusB.php
```

Requires employee email, password, order number, customer id, room number, date and time and deliver status to update the food order status and the employee that modified the status if order is in progress or delivered.

Query Params

Body raw (text)

text

```
{
  "Email": "allianad@gmail.com",
  "Password": "12345",
  "OrderNo": "3",
  "CustomerID": "3",
  "RoomNo": "2",
  "DateTime": "2021-12-25 13:30:00",
  "DeliverStatus": "inProgress"
}
```

Response:

Edited this order in the database.

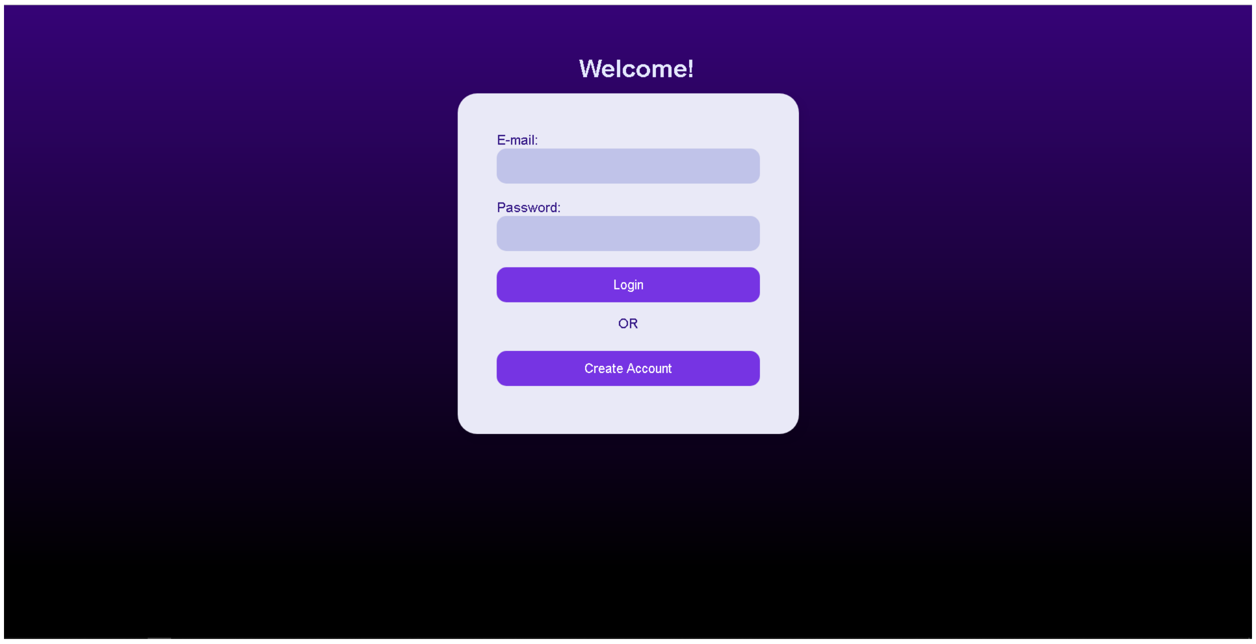
Order Number: 3

ESSN: 2

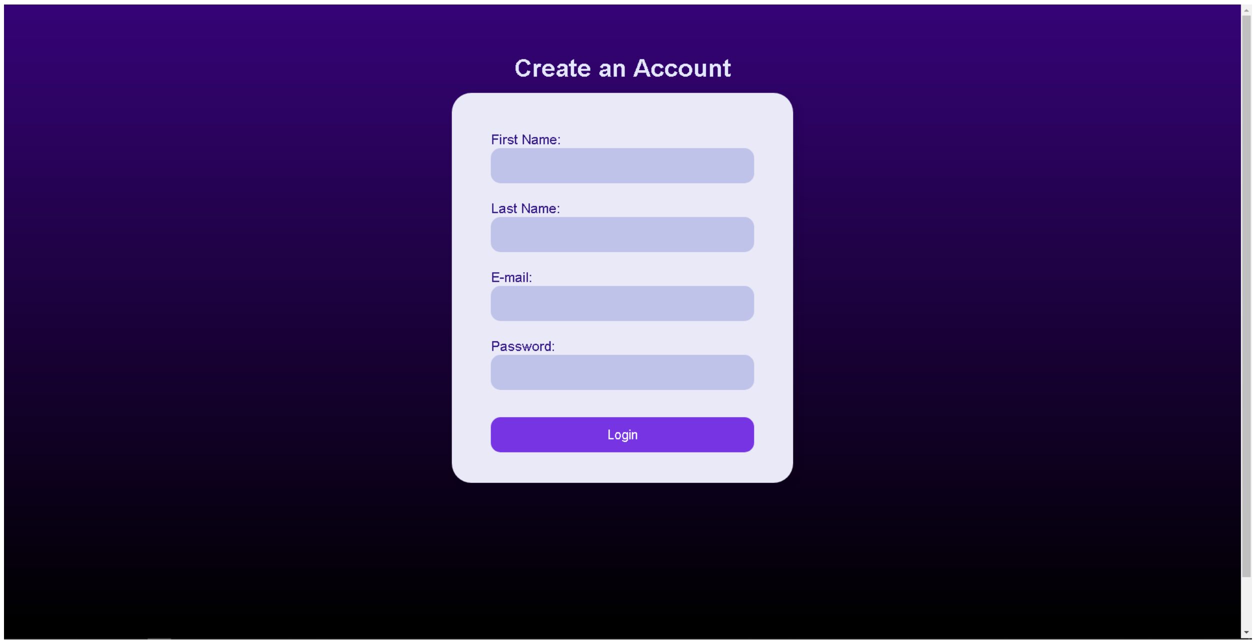
Deliver Status: In Progress

User Guide

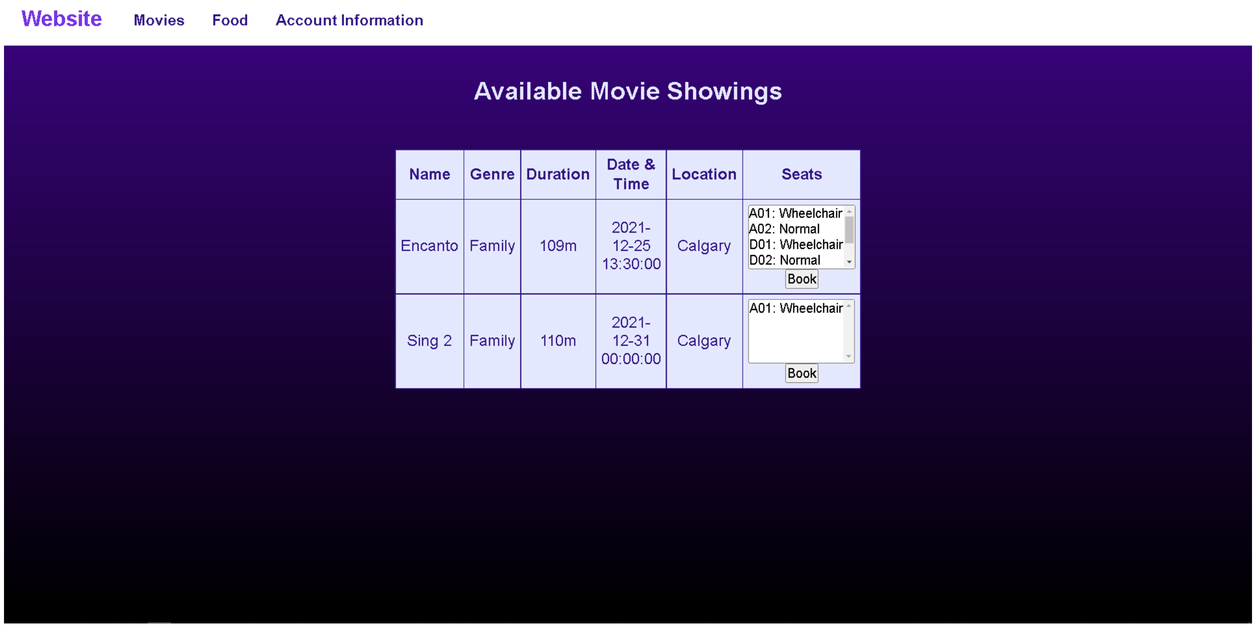
Customer Functionality:



Login Page: A page where a customer can login to their account. A login can either be successful or a failure. A successful login will bring the customer to their Movies page. A failed login will clear all information entered and refresh the page after a certain amount of time. Pressing the Create Account button will bring the user to the Create Account page.



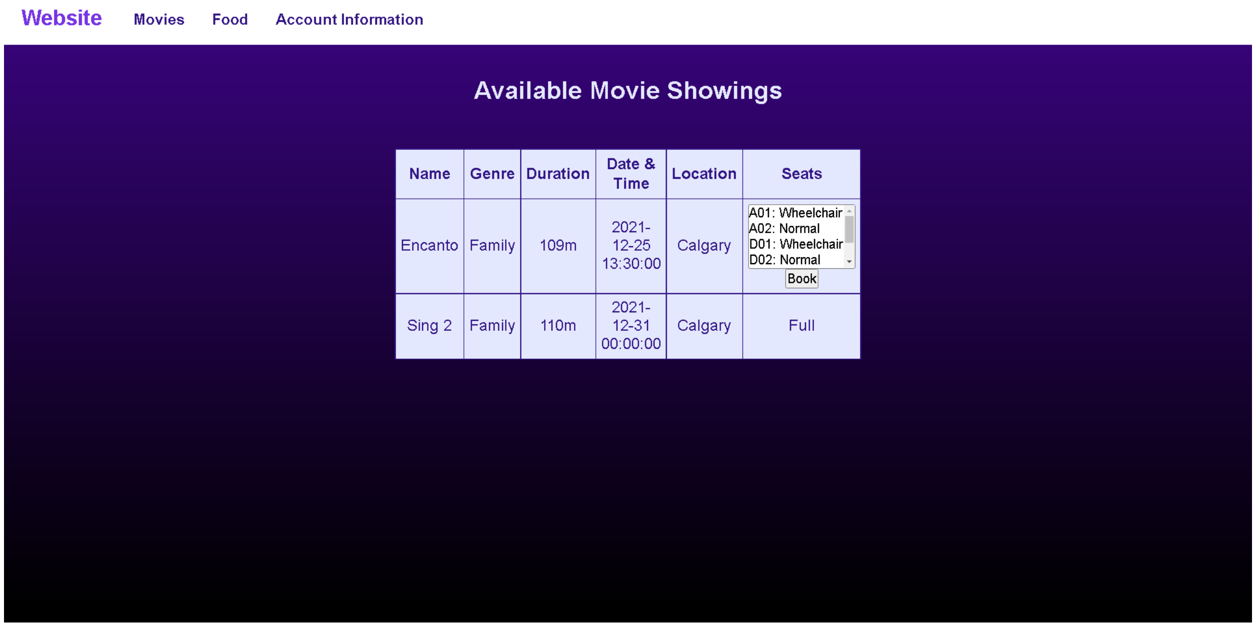
Create Account: A page where a customer can enter their information to register and create a new account. Creating a new account can either be successful or a failure. A successful account creation will register the Customer's account in the database and bring them to their Movies page. A failed account creation occurs only if any of the fields above are blank. A failed account creation will clear all information entered and refresh the page after a certain amount of time.



Available Movie Showings

Name	Genre	Duration	Date & Time	Location	Seats
Encanto	Family	109m	2021-12-25 13:30:00	Calgary	A01: Wheelchair A02: Normal D01: Wheelchair D02: Normal Book
Sing 2	Family	110m	2021-12-31 00:00:00	Calgary	A01: Wheelchair Book

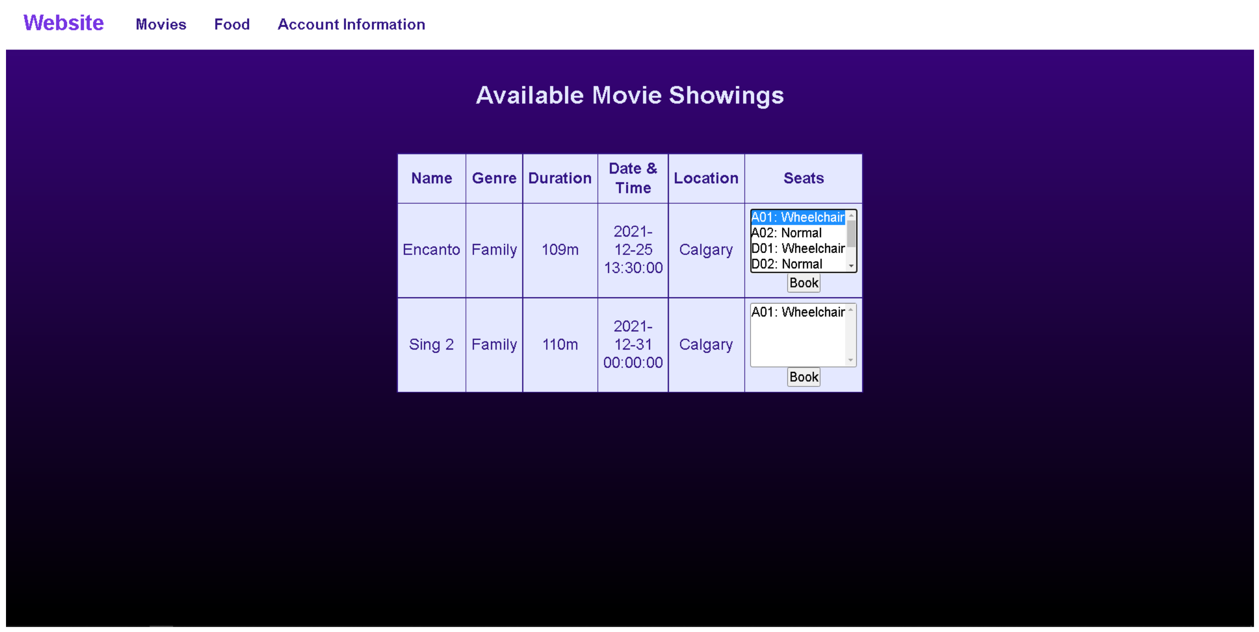
Movies page: A customer's home page. There is a taskbar at the top that holds links to different pages. Pressing the movies button will bring a user to the Movies page. Pressing the Food button will bring a customer to the Food page. Pressing the Account Information button will bring the user to the Account Information page. On the Home page, only upcoming (future) movie showings are displayed to the Customer. The customer can book seats for the Movie Showings.



Available Movie Showings

Name	Genre	Duration	Date & Time	Location	Seats
Encanto	Family	109m	2021-12-25 13:30:00	Calgary	A01: Wheelchair A02: Normal D01: Wheelchair D02: Normal Book
Sing 2	Family	110m	2021-12-31 00:00:00	Calgary	Full

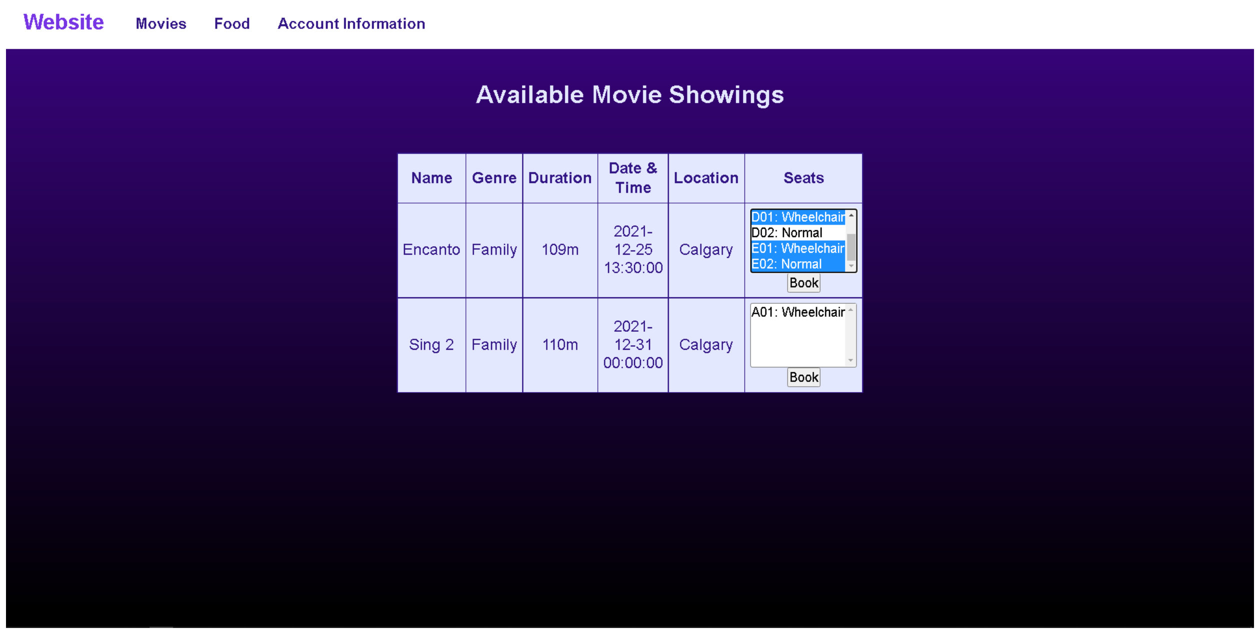
Movies Page: If there are upcoming movie showings with no seats available, Full is displayed to the customer as the movie showing is full and the customer is, therefore, unable to book seats for that movie showing.



Available Movie Showings

Name	Genre	Duration	Date & Time	Location	Seats
Encanto	Family	109m	2021-12-25 13:30:00	Calgary	A01: Wheelchair A02: Normal D01: Wheelchair D02: Normal Book
Sing 2	Family	110m	2021-12-31 00:00:00	Calgary	A01: Wheelchair Book

Movies Page: The customer is able to book one seat for a movie showing by clicking on the seat that they would like to book and then pressing the "Book" button. The scroll bar in the seats window reveals more seats. A successful booking will bring the customer to the Account Information page.

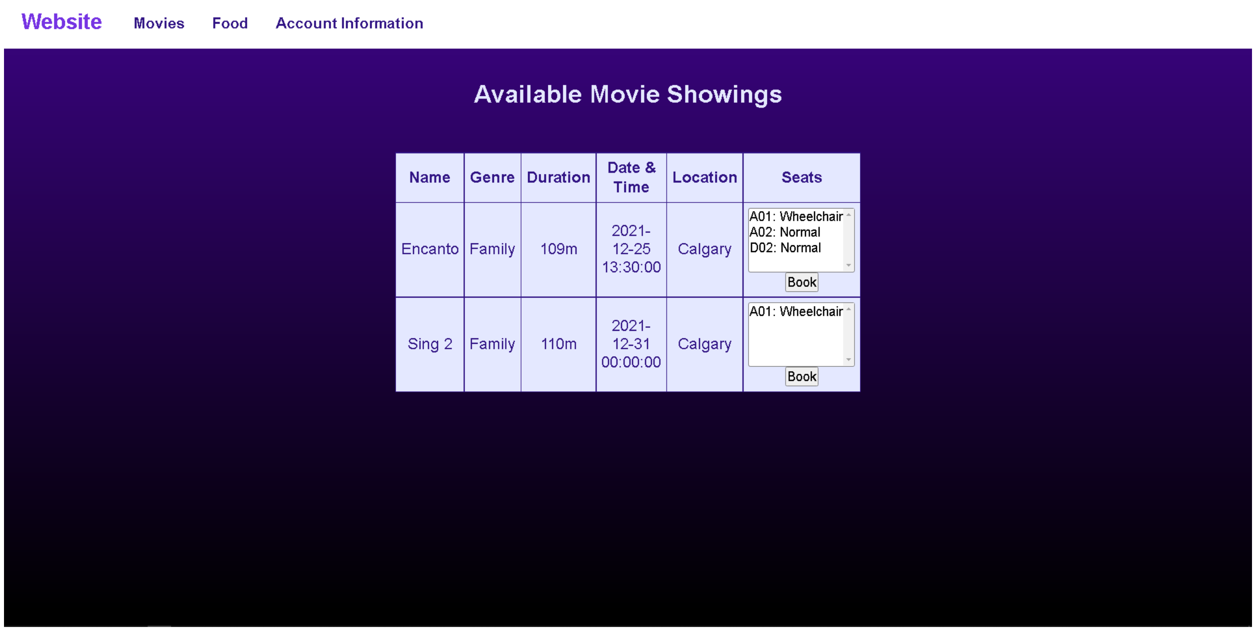


Available Movie Showings

Name	Genre	Duration	Date & Time	Location	Seats
Encanto	Family	109m	2021-12-25 13:30:00	Calgary	D01: Wheelchair D02: Normal E01: Wheelchair E02: Normal Book
Sing 2	Family	110m	2021-12-31 00:00:00	Calgary	A01: Wheelchair Book

Movies Page: The customer is able to book more than one seat by holding Ctrl and clicking on whichever seats that they would like to book. They can then book those seats by clicking the

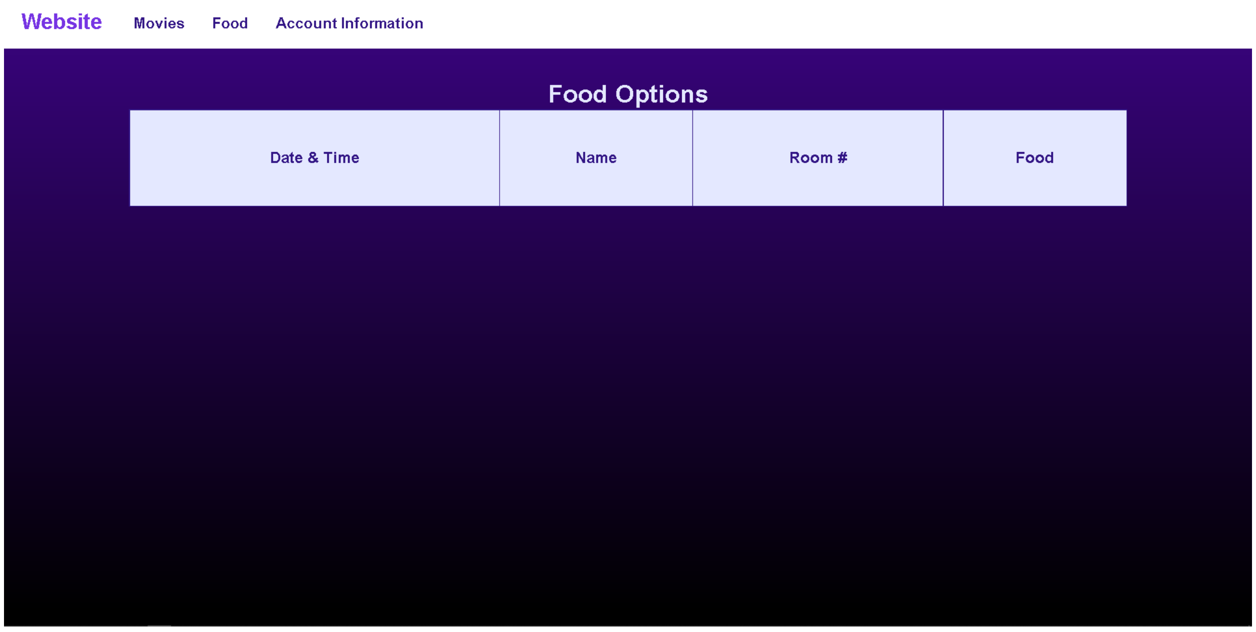
“Book” button. The scroll bar in the seats window reveals more seats. A successful booking will bring the customer to the Account Information page.



Available Movie Showings

Name	Genre	Duration	Date & Time	Location	Seats
Encanto	Family	109m	2021-12-25 13:30:00	Calgary	A01: Wheelchair A02: Normal D02: Normal Book
Sing 2	Family	110m	2021-12-31 00:00:00	Calgary	A01: Wheelchair Book

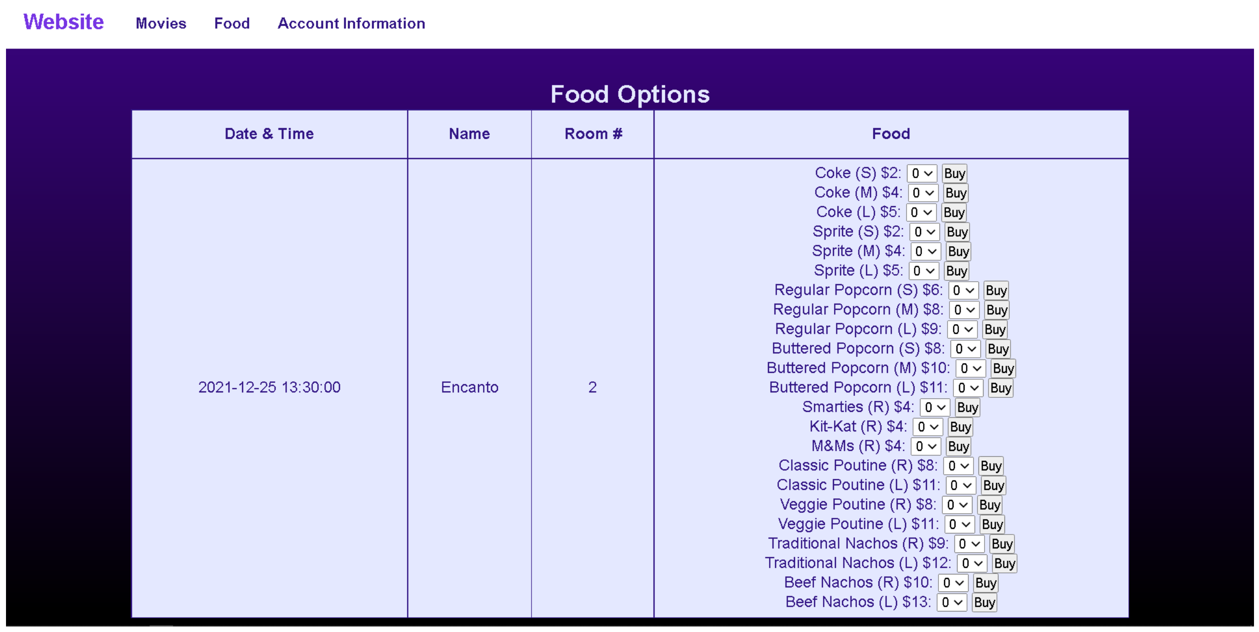
Movies Page: Let’s say the customer books the seats D01, E01, and E02. Booking these seats causes them to disappear from the available seats for that movie showing which ensures a seat cannot be double booked.



Food Options

Date & Time	Name	Room #	Food

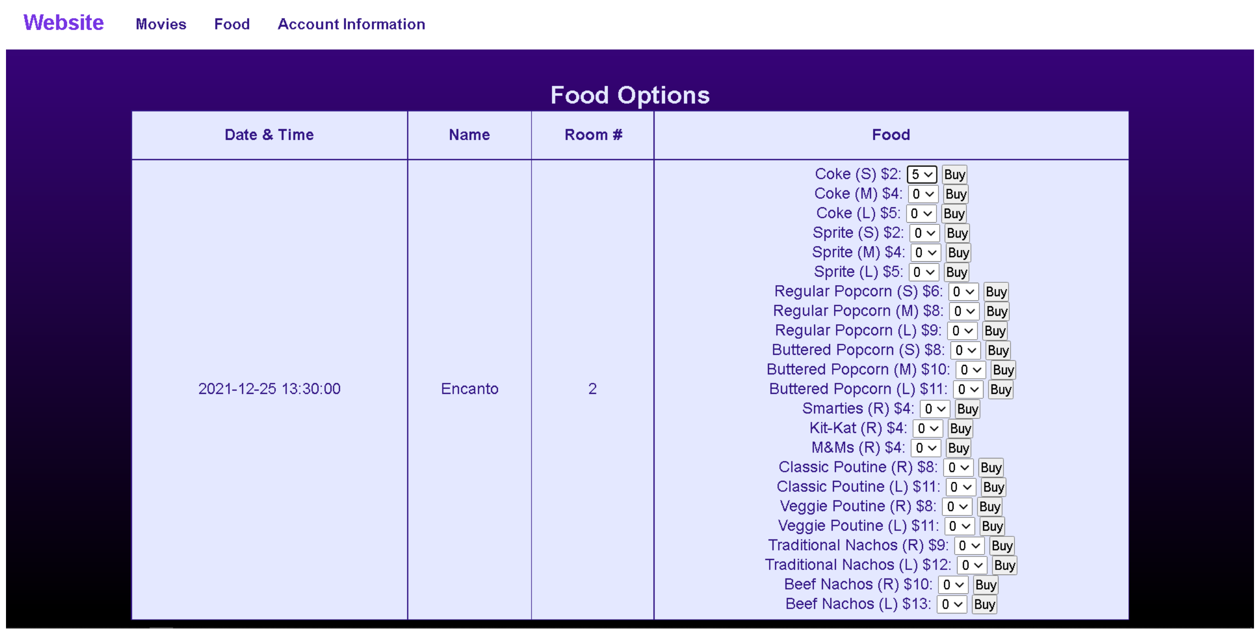
Food Page: The food page remains empty until a customer books a movie showing. This ensures that customers cannot just buy food from the cinema theatre without booking seats for a movie showing.


Website Movies Food Account Information

Food Options

Date & Time	Name	Room #	Food
2021-12-25 13:30:00	Encanto	2	Coke (S) $2: 0 Buy
			Coke (M) $4: 0 Buy
			Coke (L) $5: 0 Buy
			Sprite (S) $2: 0 Buy
			Sprite (M) $4: 0 Buy
			Sprite (L) $5: 0 Buy
			Regular Popcorn (S) $6: 0 Buy
			Regular Popcorn (M) $8: 0 Buy
			Regular Popcorn (L) $9: 0 Buy
			Buttered Popcorn (S) $8: 0 Buy
			Buttered Popcorn (M) $10: 0 Buy
			Buttered Popcorn (L) $11: 0 Buy
			Smarties (R) $4: 0 Buy
			Kit-Kat (R) $4: 0 Buy
			M&Ms (R) $4: 0 Buy
			Classic Poutine (R) $8: 0 Buy
			Classic Poutine (L) $11: 0 Buy
			Veggie Poutine (R) $8: 0 Buy
			Veggie Poutine (L) $11: 0 Buy
			Traditional Nachos (R) $9: 0 Buy
			Traditional Nachos (L) $12: 0 Buy
			Beef Nachos (R) $10: 0 Buy
			Beef Nachos (L) $13: 0 Buy

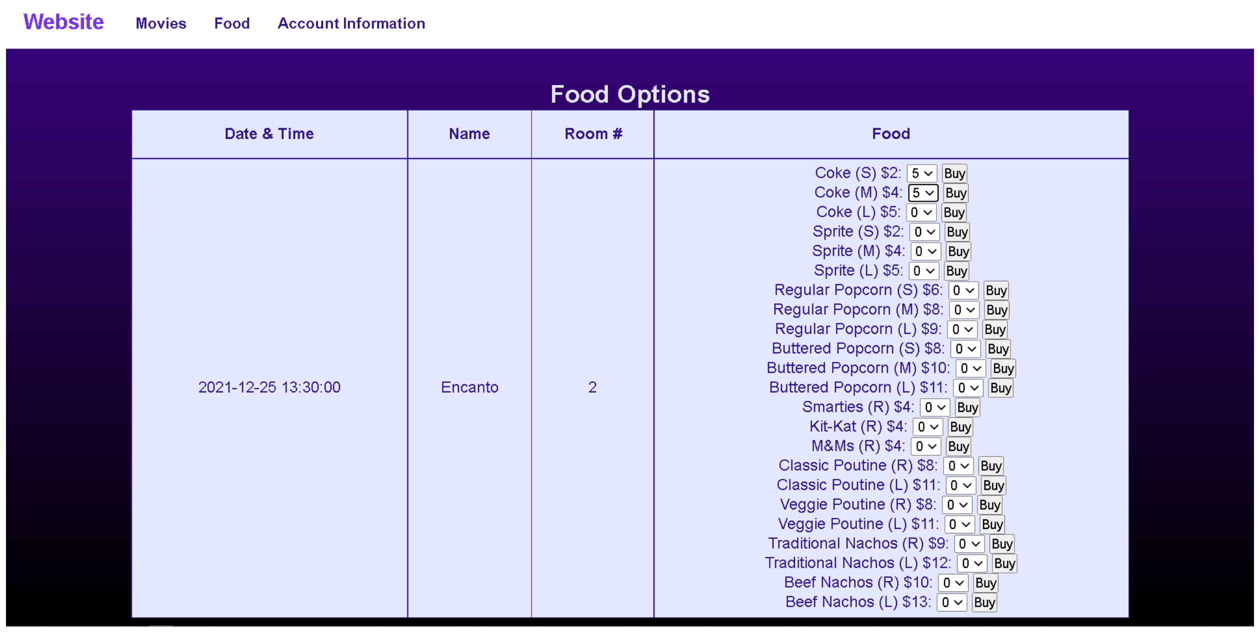
Food Page: The food options are revealed upon the successful booking of seats for a movie showing by the customer.


Website Movies Food Account Information

Food Options

Date & Time	Name	Room #	Food
2021-12-25 13:30:00	Encanto	2	Coke (S) $2: 5 Buy
			Coke (M) $4: 0 Buy
			Coke (L) $5: 0 Buy
			Sprite (S) $2: 0 Buy
			Sprite (M) $4: 0 Buy
			Sprite (L) $5: 0 Buy
			Regular Popcorn (S) $6: 0 Buy
			Regular Popcorn (M) $8: 0 Buy
			Regular Popcorn (L) $9: 0 Buy
			Buttered Popcorn (S) $8: 0 Buy
			Buttered Popcorn (M) $10: 0 Buy
			Buttered Popcorn (L) $11: 0 Buy
			Smarties (R) $4: 0 Buy
			Kit-Kat (R) $4: 0 Buy
			M&Ms (R) $4: 0 Buy
			Classic Poutine (R) $8: 0 Buy
			Classic Poutine (L) $11: 0 Buy
			Veggie Poutine (R) $8: 0 Buy
			Veggie Poutine (L) $11: 0 Buy
			Traditional Nachos (R) $9: 0 Buy
			Traditional Nachos (L) $12: 0 Buy
			Beef Nachos (R) $10: 0 Buy
			Beef Nachos (L) $13: 0 Buy

Food Page: Each food option has a dropdown that allows for a user to buy 0-5 of that specific food. To successfully buy a food and create a food order, the quantity for that food must be greater than 0 and the respective "Buy" button must be clicked. For example, 5 of the Coke (S) $2 are being bought. If a button other than the one next to the Coke (S) $2 were to be clicked, it would result in a failure to create a food order. Only one type of food can be bought at a time. A successful creation of a food order will bring the customer to the Account Information page.

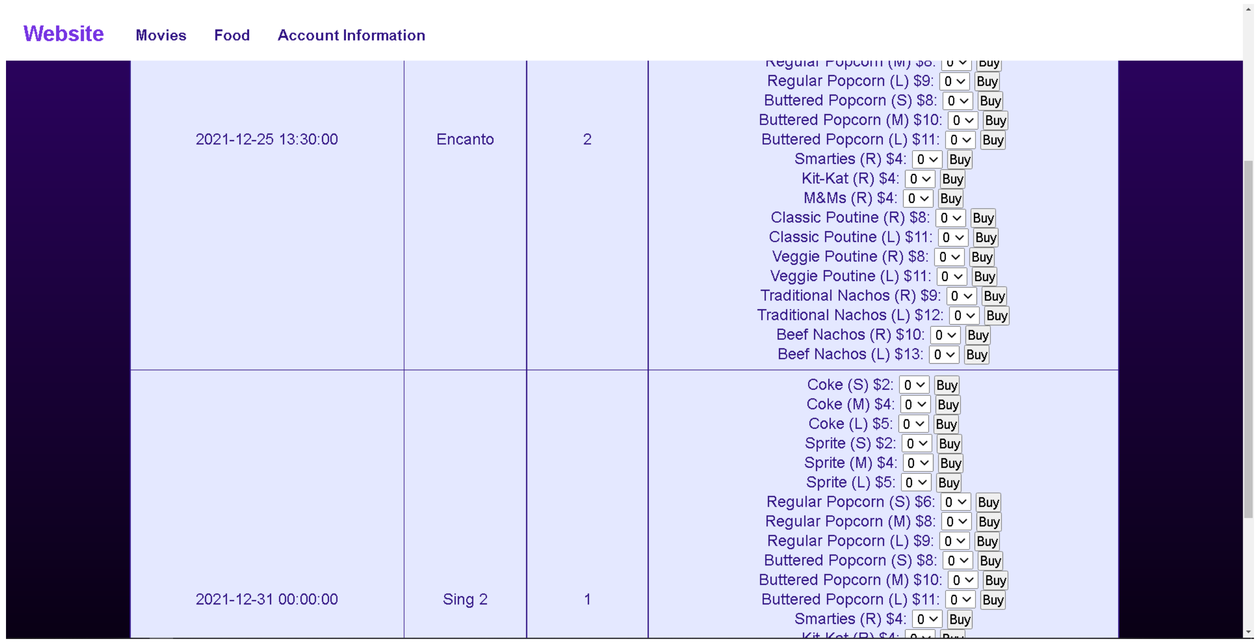


Website Movies Food Account Information

Food Options

Date & Time	Name	Room #	Food
2021-12-25 13:30:00	Encanto	2	Coke (S) $2: 5 Buy Coke (M) $4: 5 Buy Coke (L) $5: 0 Buy Sprite (S) $2: 0 Buy Sprite (M) $4: 0 Buy Sprite (L) $5: 0 Buy Regular Popcorn (S) $6: 0 Buy Regular Popcorn (M) $8: 0 Buy Regular Popcorn (L) $9: 0 Buy Buttered Popcorn (S) $8: 0 Buy Buttered Popcorn (M) $10: 0 Buy Buttered Popcorn (L) $11: 0 Buy Smarties (R) $4: 0 Buy Kit-Kat (R) $4: 0 Buy M&Ms (R) $4: 0 Buy Classic Poutine (R) $8: 0 Buy Classic Poutine (L) $11: 0 Buy Veggie Poutine (R) $8: 0 Buy Veggie Poutine (L) $11: 0 Buy Traditional Nachos (R) $9: 0 Buy Traditional Nachos (L) $12: 0 Buy Beef Nachos (R) $10: 0 Buy Beef Nachos (L) $13: 0 Buy

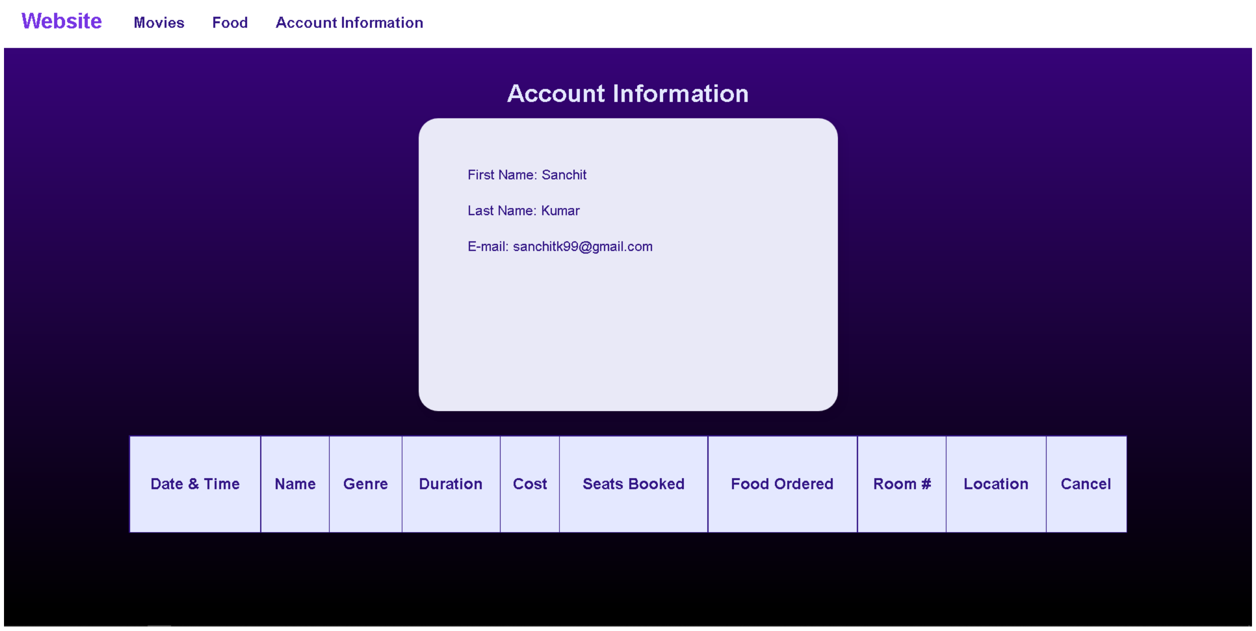
Food Page: Here, two food options are filled in at the same time. The customer is trying to buy 5 of both Coke (S) $2 and Coke (M) $4. However, if the customer clicks the buy button next to the Coke (S) $2, then only the Coke (S) $2 will be considered as being bought by the user. Therefore, it is not possible to buy more than one type of food at a time. A user can come back to the Food page to buy other food for the same movie showing. As long as it is for the same movie showing, it will be considered to be a part of the customer's food order. A successful creation of a food order will bring the customer to the Account Information page.



Website Movies Food Account Information

2021-12-25 13:30:00	Encanto	2	Regular Popcorn (L) $9: 0 Buy Buttered Popcorn (S) $8: 0 Buy Buttered Popcorn (M) $10: 0 Buy Buttered Popcorn (L) $11: 0 Buy Smarties (R) $4: 0 Buy Kit-Kat (R) $4: 0 Buy M&Ms (R) $4: 0 Buy Classic Poutine (R) $8: 0 Buy Classic Poutine (L) $11: 0 Buy Veggie Poutine (R) $8: 0 Buy Veggie Poutine (L) $11: 0 Buy Traditional Nachos (R) $9: 0 Buy Traditional Nachos (L) $12: 0 Buy Beef Nachos (R) $10: 0 Buy Beef Nachos (L) $13: 0 Buy
2021-12-31 00:00:00	Sing 2	1	Coke (S) $2: 0 Buy Coke (M) $4: 0 Buy Coke (L) $5: 0 Buy Sprite (S) $2: 0 Buy Sprite (M) $4: 0 Buy Sprite (L) $5: 0 Buy Regular Popcorn (S) $6: 0 Buy Regular Popcorn (M) $8: 0 Buy Regular Popcorn (L) $9: 0 Buy Buttered Popcorn (S) $8: 0 Buy Buttered Popcorn (M) $10: 0 Buy Buttered Popcorn (L) $11: 0 Buy Smarties (R) $4: 0 Buy

Food Page: If the customer has seats booked for multiple different movie showings, the food options are displayed for each booking. A customer can then order food for each different movie showing and a new food order is created for that specific movie showing.


Website   Movies   Food   Account Information

Account Information

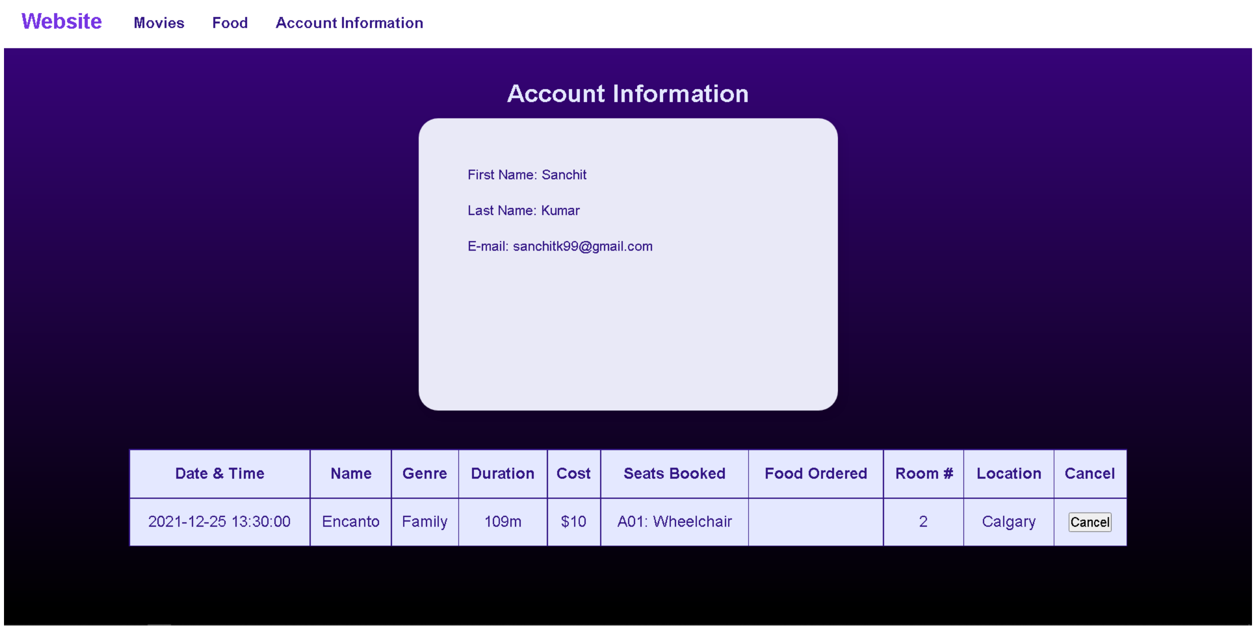
First Name: Sanchit

Last Name: Kumar

E-mail: sanchitk99@gmail.com

Date & Time	Name	Genre	Duration	Cost	Seats Booked	Food Ordered	Room #	Location	Cancel

Account Information Page: The account information of the customer is displayed. At the bottom, each of the bookings of the customer are displayed. Since the customer has no bookings, there is no information for bookings displayed.


Website   Movies   Food   Account Information

Account Information

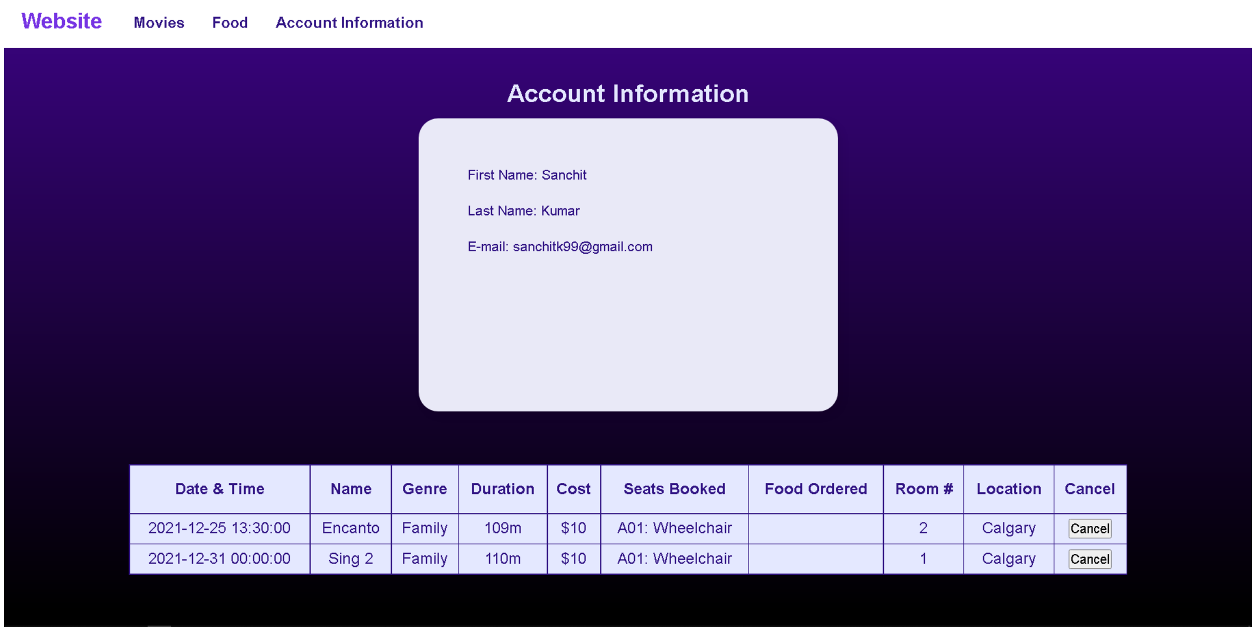
First Name: Sanchit

Last Name: Kumar

E-mail: sanchitk99@gmail.com

Date & Time	Name	Genre	Duration	Cost	Seats Booked	Food Ordered	Room #	Location	Cancel
2021-12-25 13:30:00	Encanto	Family	109m	$10	A01: Wheelchair		2	Calgary	Cancel

Account Information Page: Let's say a customer books the A01 seat for a movie showing on the 25th of December 2021 for the movie Encanto and doesn't order any food for the movie showing. Then, the Date and Time, name, genre, duration, and the location and room number of the movie are displayed to the user. At the same time, the total cost of the booking is displayed to the customer ($10 per seat). The food ordered column is left empty as the customer has not ordered any food for the movie showing. The customer is also given a “Cancel” button to cancel the movie showing at any time.



Website Movies Food Account Information

Account Information

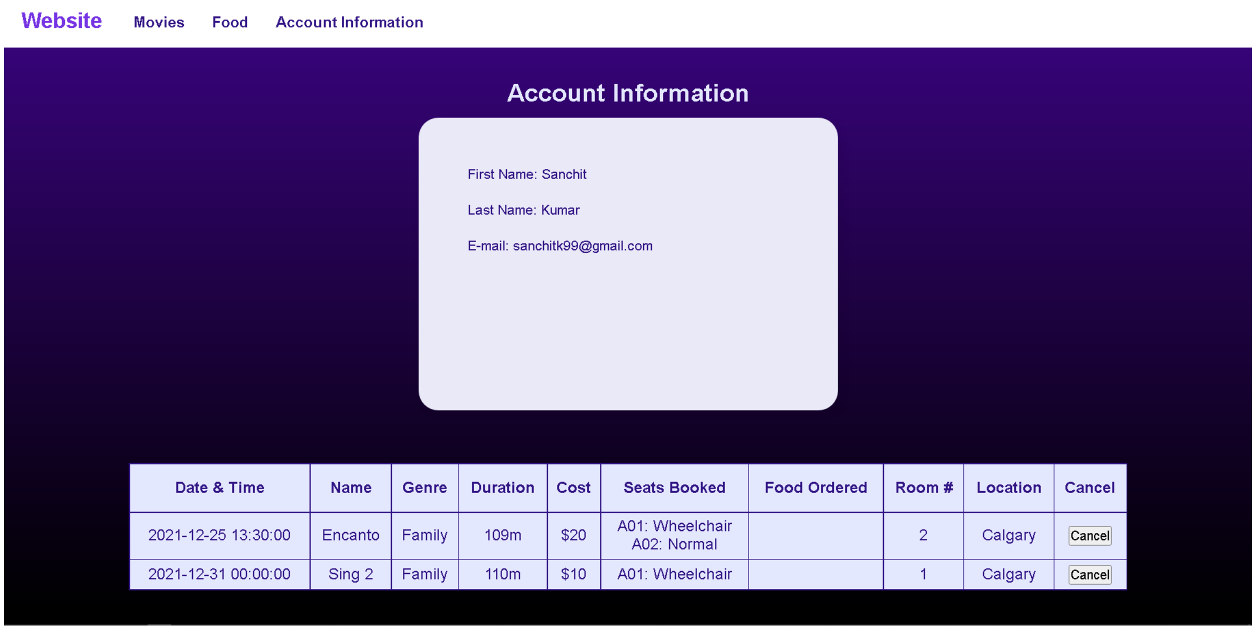
First Name: Sanchit

Last Name: Kumar

E-mail: sanchitk99@gmail.com

Date & Time	Name	Genre	Duration	Cost	Seats Booked	Food Ordered	Room #	Location	Cancel
2021-12-25 13:30:00	Encanto	Family	109m	$10	A01: Wheelchair		2	Calgary	Cancel
2021-12-31 00:00:00	Sing 2	Family	110m	$10	A01: Wheelchair		1	Calgary	Cancel

Account Information Page: Let’s say the customer also books the A01 seat for a movie showing on the 31st of December 2021 for the movie Sing 2 and doesn’t order any food for the movie. The information of the second movie is also displayed in the same way.



Website Movies Food Account Information

Account Information

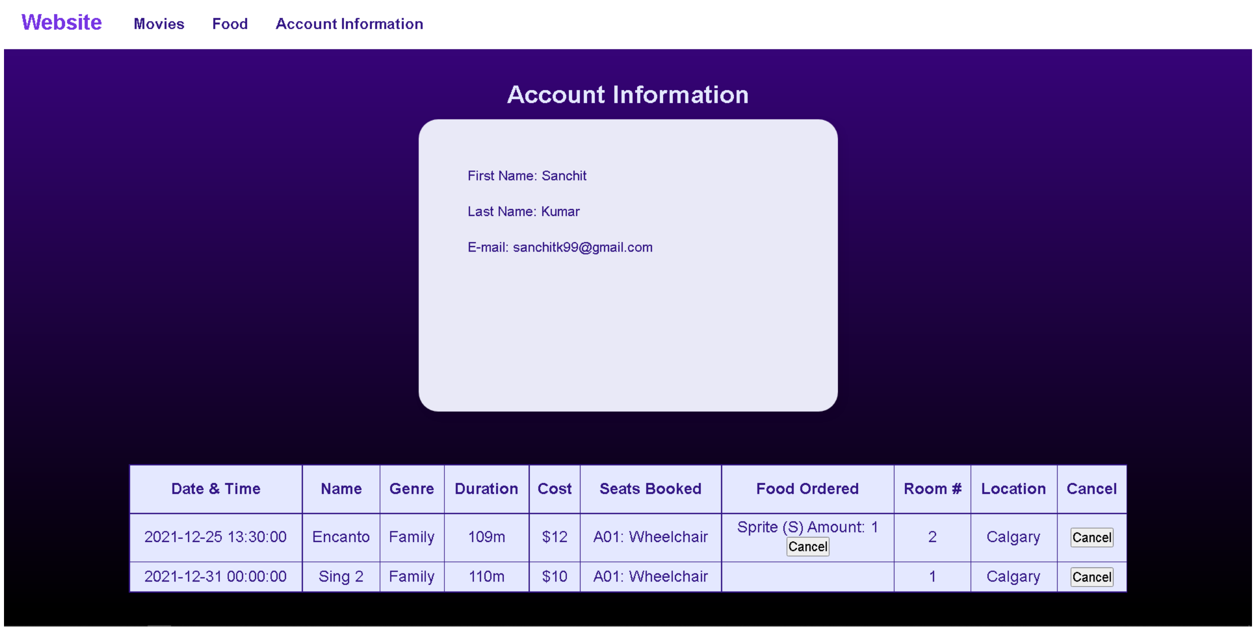
First Name: Sanchit

Last Name: Kumar

E-mail: sanchitk99@gmail.com

Date & Time	Name	Genre	Duration	Cost	Seats Booked	Food Ordered	Room #	Location	Cancel
2021-12-25 13:30:00	Encanto	Family	109m	$20	A01: Wheelchair A02: Normal		2	Calgary	Cancel
2021-12-31 00:00:00	Sing 2	Family	110m	$10	A01: Wheelchair		1	Calgary	Cancel

Account Information Page: Let’s say the customer also wants to book the A02 seat for the movie showing on the 25th of December 2021 for the movie Encanto. The newly booked seat is added to the previous booking that the customer had made for the same movie showing. The seats booked column is updated to show every seat the customer has booked for the movie showing. The cost of the booking is also updated to $20 ($10 per seat).



Account Information

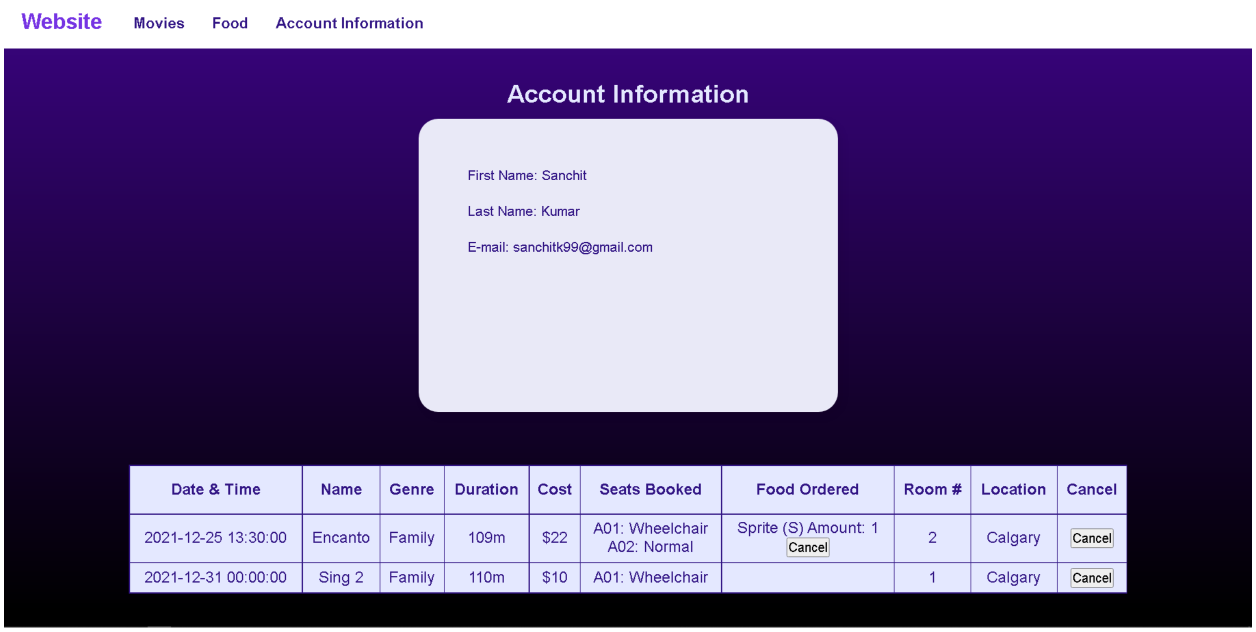
First Name: Sanchit

Last Name: Kumar

E-mail: sanchitk99@gmail.com

Date & Time	Name	Genre	Duration	Cost	Seats Booked	Food Ordered	Room #	Location	Cancel
2021-12-25 13:30:00	Encanto	Family	109m	$12	A01: Wheelchair	Sprite (S) Amount: 1 Cancel	2	Calgary	Cancel
2021-12-31 00:00:00	Sing 2	Family	110m	$10	A01: Wheelchair		1	Calgary	Cancel

Account Information: The customer can decide to buy food for the bookings anytime before the movie showing by clicking on the Food button and moving to the Food page. Let’s say the user decides to buy one Sprite (S) $2. Then, the food that they have decided to buy for the booking is displayed in the Food Ordered column. A cancel button also appears that allows the user to just cancel the food order. At the same time, the cost of the booking is also updated to $12 ($10 for the seat and $2 for the Sprite (S)).



Account Information

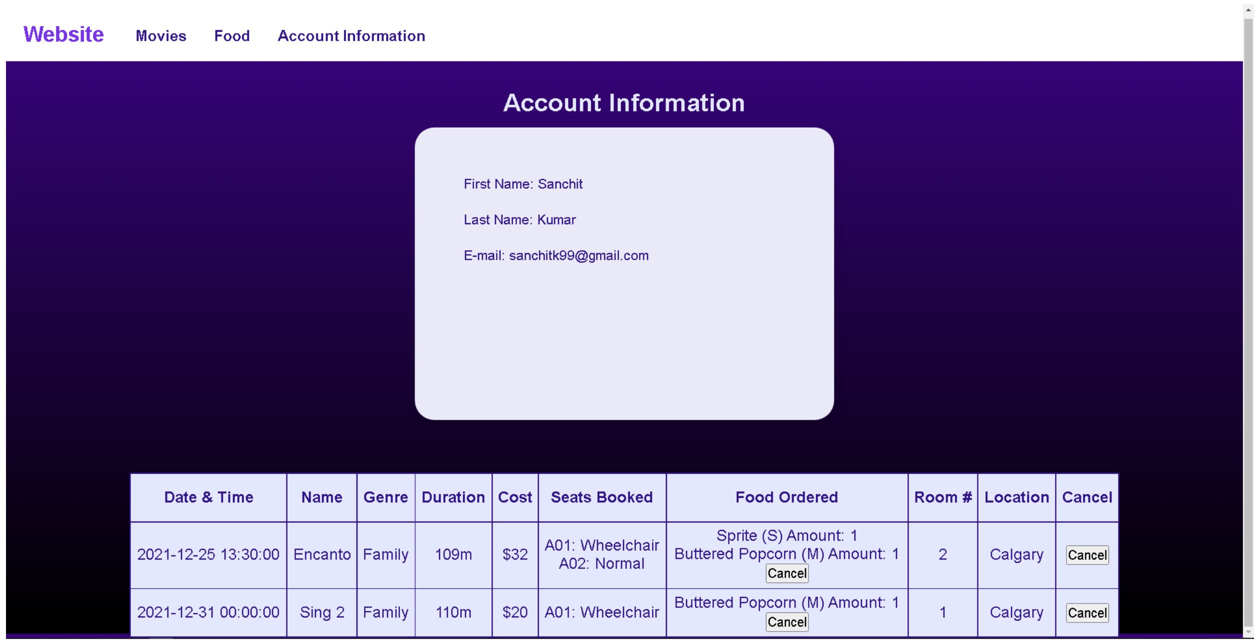
First Name: Sanchit

Last Name: Kumar

E-mail: sanchitk99@gmail.com

Date & Time	Name	Genre	Duration	Cost	Seats Booked	Food Ordered	Room #	Location	Cancel
2021-12-25 13:30:00	Encanto	Family	109m	$22	A01: Wheelchair A02: Normal	Sprite (S) Amount: 1 Cancel	2	Calgary	Cancel
2021-12-31 00:00:00	Sing 2	Family	110m	$10	A01: Wheelchair		1	Calgary	Cancel

Account Information Page: The customer can reserve seats for a movie showing at any time before that movie showing starts. If the customer has already ordered food for the movie showing, they can still book more seats or more food for that same movie showing. The seats booked and the cost of the booking will be updated accordingly.



Account Information

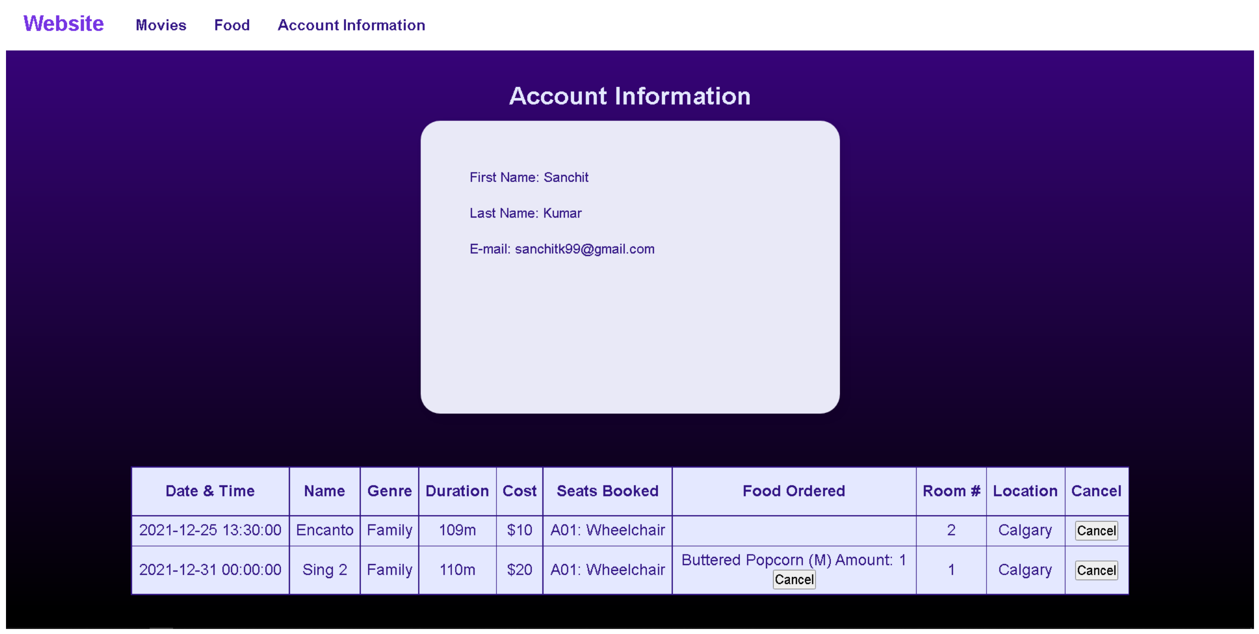
First Name: Sanchit

Last Name: Kumar

E-mail: sanchitk99@gmail.com

Date & Time	Name	Genre	Duration	Cost	Seats Booked	Food Ordered	Room #	Location	Cancel
2021-12-25 13:30:00	Encanto	Family	109m	$32	A01: Wheelchair A02: Normal	Sprite (S) Amount: 1 Buttered Popcorn (M) Amount: 1 Cancel	2	Calgary	Cancel
2021-12-31 00:00:00	Sing 2	Family	110m	$20	A01: Wheelchair	Buttered Popcorn (M) Amount: 1 Cancel	1	Calgary	Cancel

Account Information Page: Now, let's say that the customer decides to buy one Buttered Popcorn (M) $10 for each of the bookings. The Buttered Popcorn (M) is added to the food order of the 1st booking and a new food order is created for the 2nd booking since there was no food ordered previously for the 2nd booking. The cost of both is updated as well.



Account Information

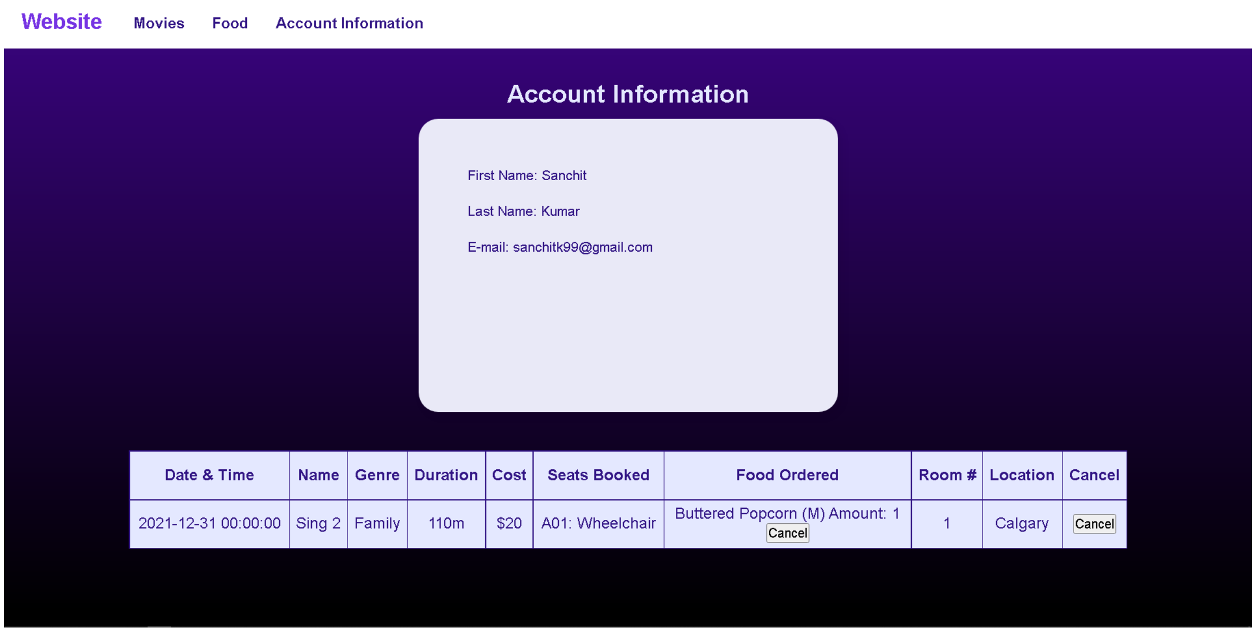
First Name: Sanchit

Last Name: Kumar

E-mail: sanchitk99@gmail.com

Date & Time	Name	Genre	Duration	Cost	Seats Booked	Food Ordered	Room #	Location	Cancel
2021-12-25 13:30:00	Encanto	Family	109m	$10	A01: Wheelchair		2	Calgary	Cancel
2021-12-31 00:00:00	Sing 2	Family	110m	$20	A01: Wheelchair	Buttered Popcorn (M) Amount: 1 Cancel	1	Calgary	Cancel

Account Information Page: Let’s say the customer wants to cancel the food order of the 1st booking. This is the result after clicking on the “Cancel” button in the Food Ordered column. The food order is deleted from this customer’s booking. The Food Ordered column is now empty (as there is now no food ordered for the 1st booking by this customer) and the cost is updated as well to now only be the cost of the seats (as there is no food ordered) rather than the cost of the seats and the cost of the food ordered. The information of the booking itself is retained, only the information of the food ordered is deleted.



Account Information

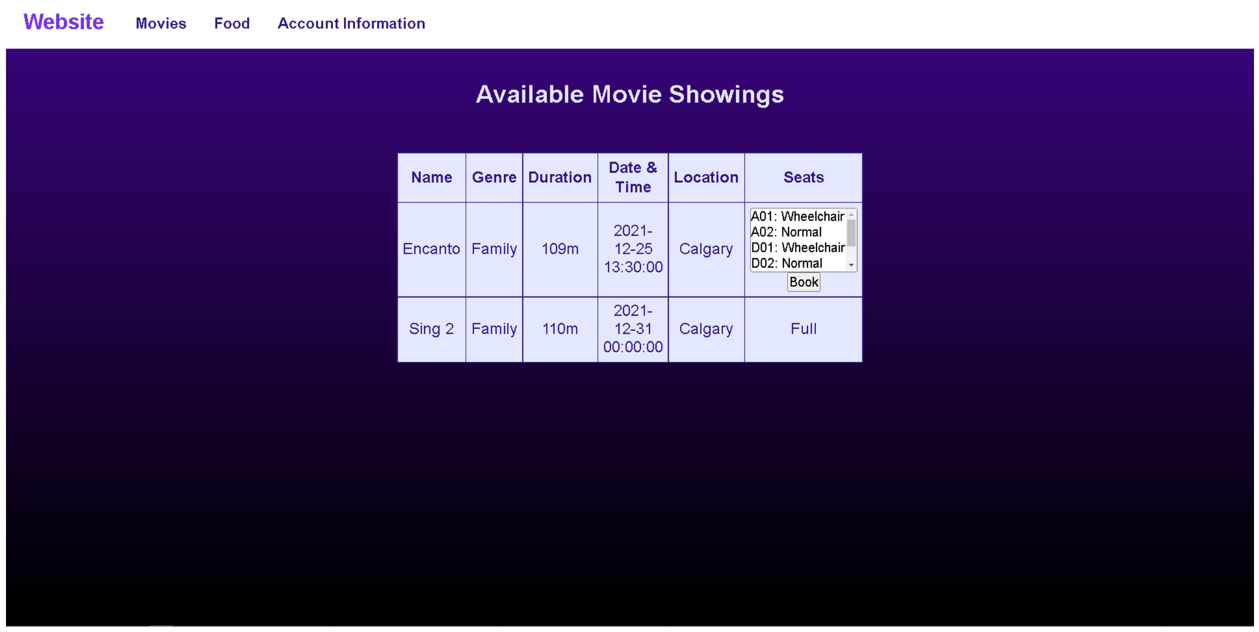
First Name: Sanchit

Last Name: Kumar

E-mail: sanchitk99@gmail.com

Date & Time	Name	Genre	Duration	Cost	Seats Booked	Food Ordered	Room #	Location	Cancel
2021-12-31 00:00:00	Sing 2	Family	110m	$20	A01: Wheelchair	Buttered Popcorn (M) Amount: 1 Cancel	1	Calgary	Cancel

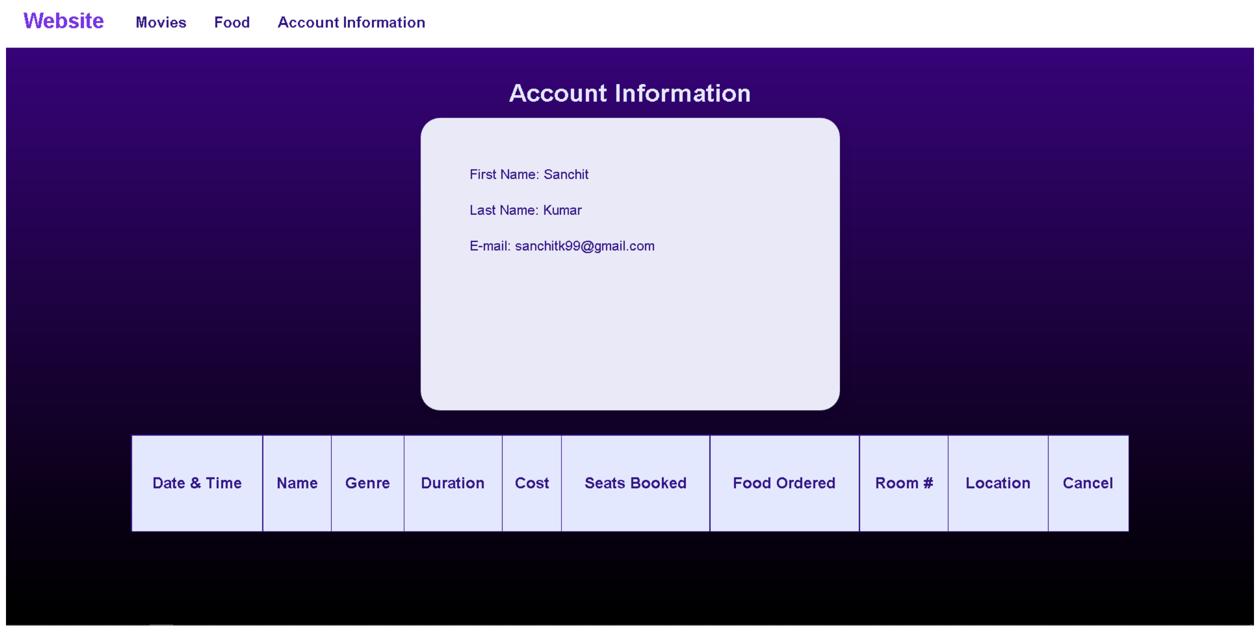
Account Information Page: Let’s say the customer wants to cancel the booking for the Encanto film on the 25th of December 2021. This is the result after clicking the “Cancel” button in the Cancel column of the booking. It removed that booking from the customer’s bookings. At the same time, because the customer has cancelled the booking ahead of the movie showing, they do not have to pay for the seats that they had previously booked and the seats that they had previously booked are now available to be booked once again by any customer.



Available Movie Showings

Name	Genre	Duration	Date & Time	Location	Seats
Encanto	Family	109m	2021-12-25 13:30:00	Calgary	A01: Wheelchair A02: Normal D01: Wheelchair D02: Normal
Sing 2	Family	110m	2021-12-31 00:00:00	Calgary	Full

Here, the A01 seat is displayed as an available seat once again in the Movies page for the Encanto film as the customer has decided that they wanted to cancel their booking for the Encanto film.



Account Information

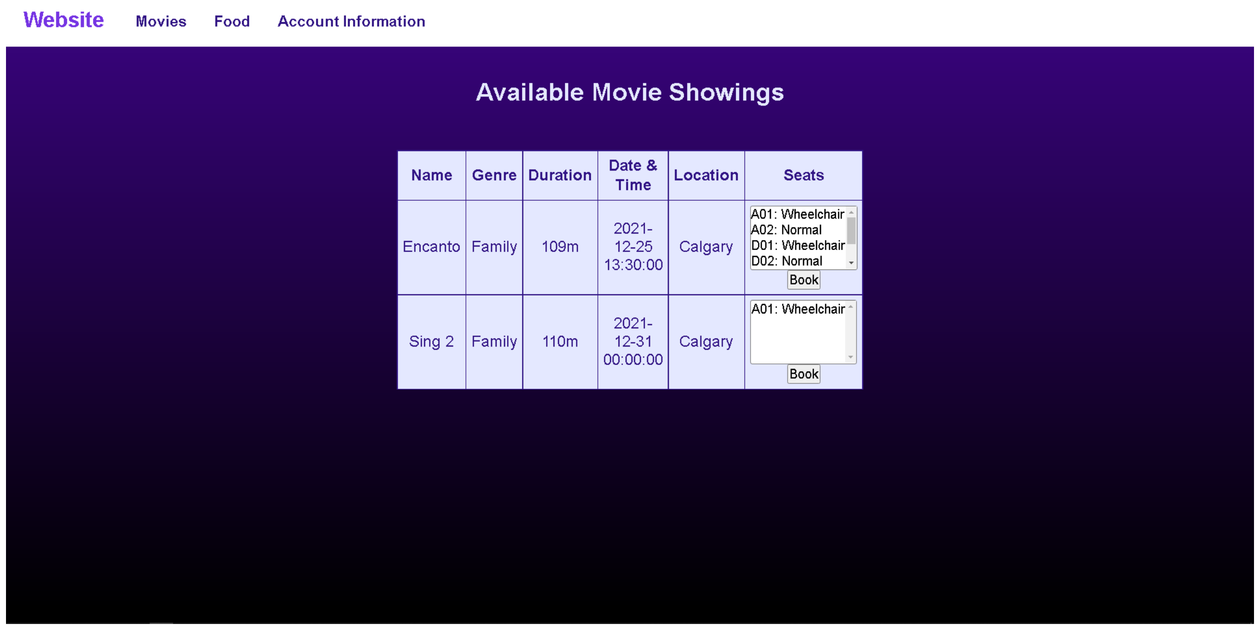
First Name: Sanchit

Last Name: Kumar

E-mail: sanchitk99@gmail.com

Date & Time	Name	Genre	Duration	Cost	Seats Booked	Food Ordered	Room #	Location	Cancel

Account Information Page: The customer can also decide to cancel the entire booking along with the food ordered instead of first cancelling the food ordered and then cancelling the booking. By clicking the “Cancel” button in the Cancel column, the booking that they wish to cancel will be removed from their bookings. At the same time, because the customer has cancelled the booking ahead of the movie showing, their food ordered for this booking will also be cancelled and the customer will not have to pay for the seats that they had booked or the food that they had ordered for the booking that they decided to cancel. The seats that they had previously booked for the movie showing will now be available to be booked once again by any customers.

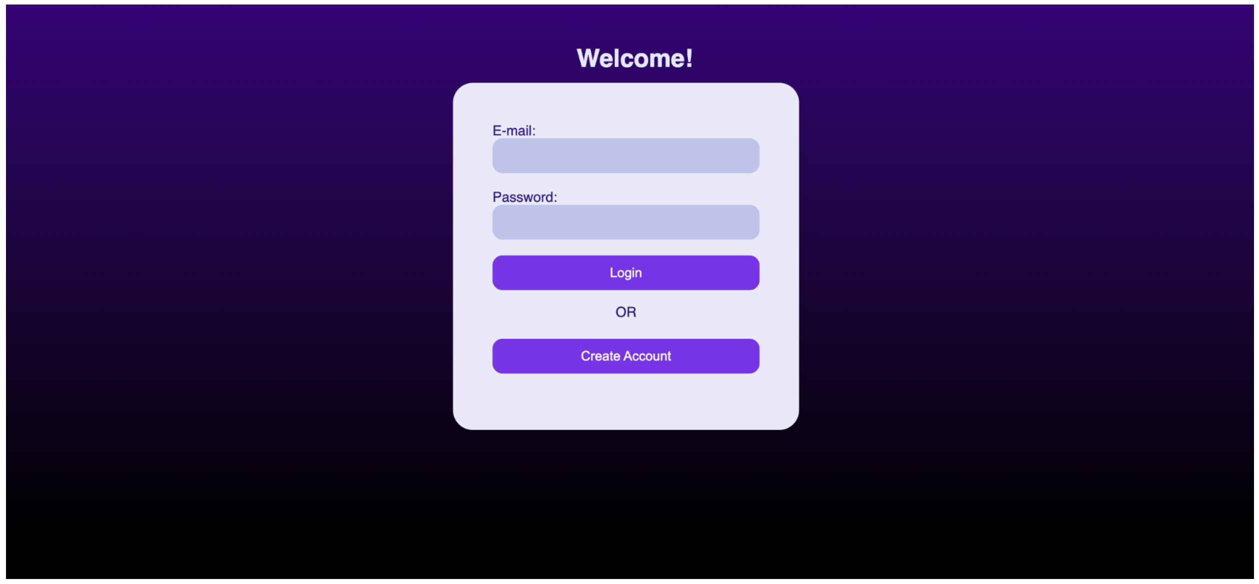

Website Movies Food Account Information

Available Movie Showings

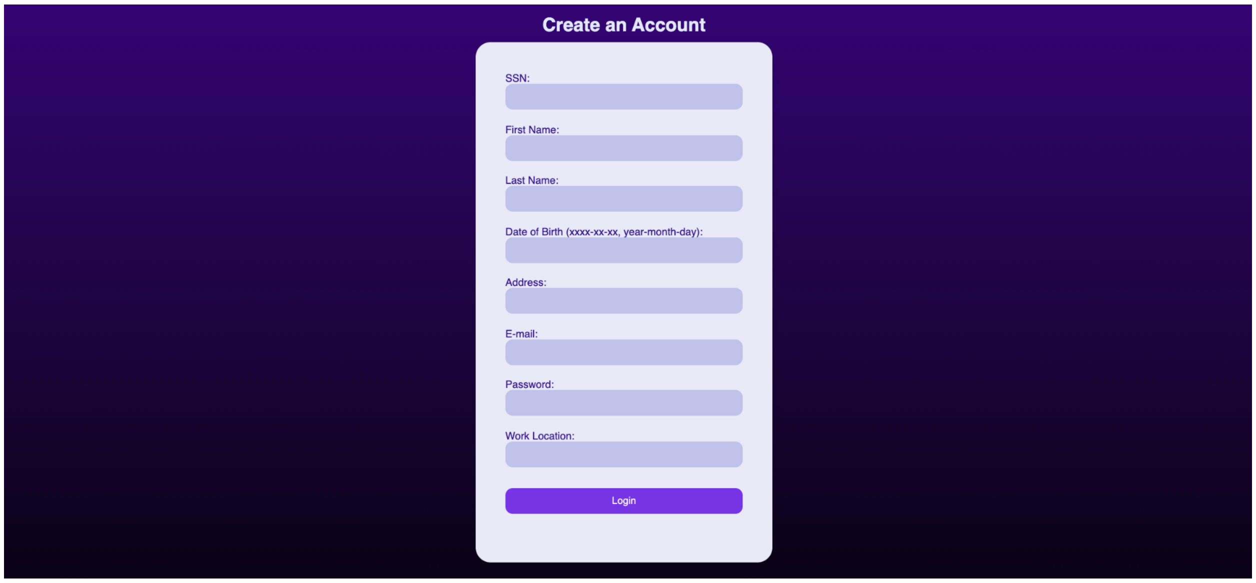
Name	Genre	Duration	Date & Time	Location	Seats
Encanto	Family	109m	2021-12-25 13:30:00	Calgary	A01: Wheelchair A02: Normal D01: Wheelchair D02: Normal Book
Sing 2	Family	110m	2021-12-31 00:00:00	Calgary	A01: Wheelchair Book

Here, the A01 seat is displayed as an available seat once again in the Movies page for the Sing 2 film as the customer has decided that they wanted to cancel their booking for the Sing 2 film.

Employee Functionality:



Login Page: Similar to the customer login page, an employee enters their email and password to log in. If the employee enters their information incorrectly, an error message will appear. If the employee is not registered yet they can click on the 'Create Account' button to be redirected to the registration page.


Create an Account

SSN:

First Name:

Last Name:

Date of Birth (xxxx-xx-xx, year-month-day):

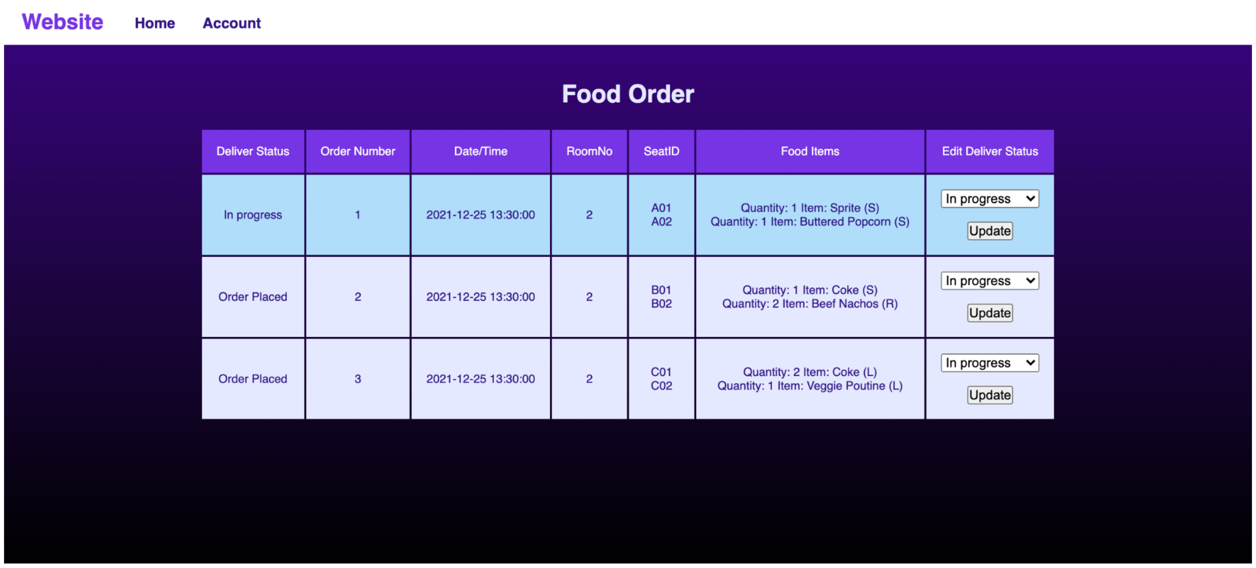
Address:

E-mail:

Password:

Work Location:

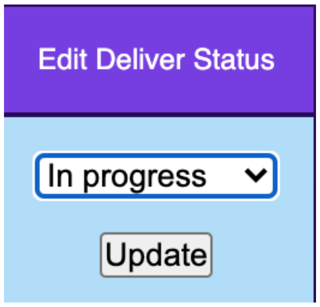
Registration Page: An employee must enter all fields, input a unique email and enter an existing work location or an error message will appear. The work location will determine who their manager will be and the theatre id of the theatre they work in. Once they correctly fill the form and click on the 'Login' button, employees will be redirected to the home page.

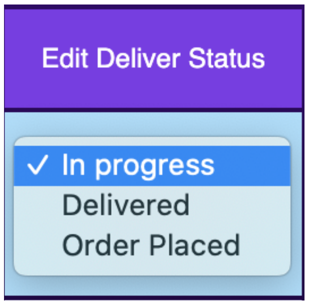


Food Order

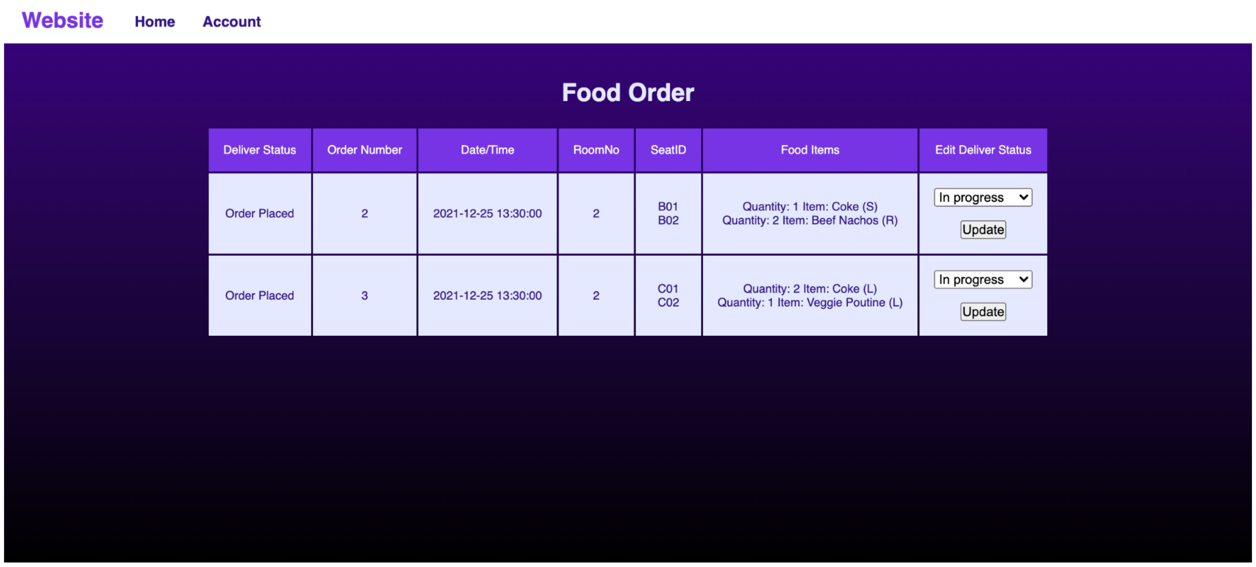
Deliver Status	Order Number	Date/Time	RoomNo	SeatID	Food Items	Edit Deliver Status
In progress	1	2021-12-25 13:30:00	2	A01 A02	Quantity: 1 Item: Sprite (S) Quantity: 1 Item: Buttered Popcorn (S)	In progress Update
Order Placed	2	2021-12-25 13:30:00	2	B01 B02	Quantity: 1 Item: Coke (S) Quantity: 2 Item: Beef Nachos (R)	In progress Update
Order Placed	3	2021-12-25 13:30:00	2	C01 C02	Quantity: 2 Item: Coke (L) Quantity: 1 Item: Veggie Poutine (L)	In progress Update

Employee Home Page: This page shows the list of food orders to be completed for the employee's theatre location. The row in blue indicates that the food order is being completed by their coworker. Each row tells the employee what needs to be delivered, where and when.





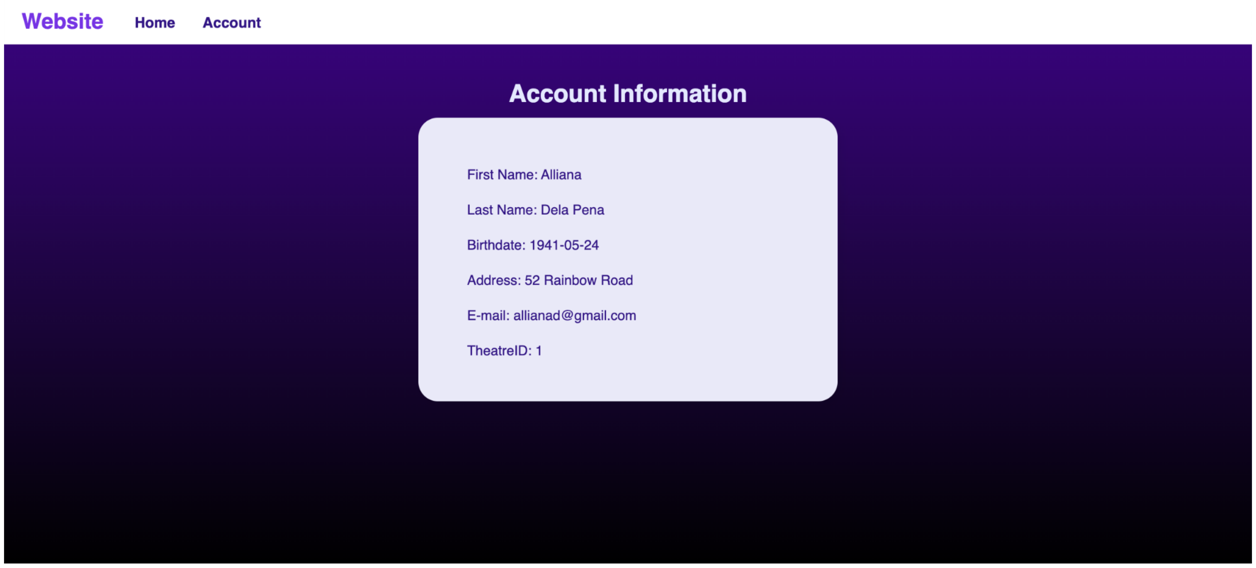
In the Edit Deliver Status column the employee can let their coworkers know if they delivered or are currently working on completing a food order. They click on the drop down menu and set it to the appropriate delivery status. Next, the employee must click the Update button to save the changes.



Food Order

Deliver Status	Order Number	Date/Time	RoomNo	SeatID	Food Items	Edit Deliver Status
Order Placed	2	2021-12-25 13:30:00	2	B01 B02	Quantity: 1 Item: Coke (S) Quantity: 2 Item: Beef Nachos (R)	In progress Update
Order Placed	3	2021-12-25 13:30:00	2	C01 C02	Quantity: 2 Item: Coke (L) Quantity: 1 Item: Veggie Poutine (L)	In progress Update

Employee Home Page: Once an order is completed, the employee edits the deliver status to 'Delivered'. After clicking on the 'Update' button, the order should be removed from the page.


Website
Home
Account

Account Information

First Name: Alliana

Last Name: Dela Pena

Birthdate: 1941-05-24

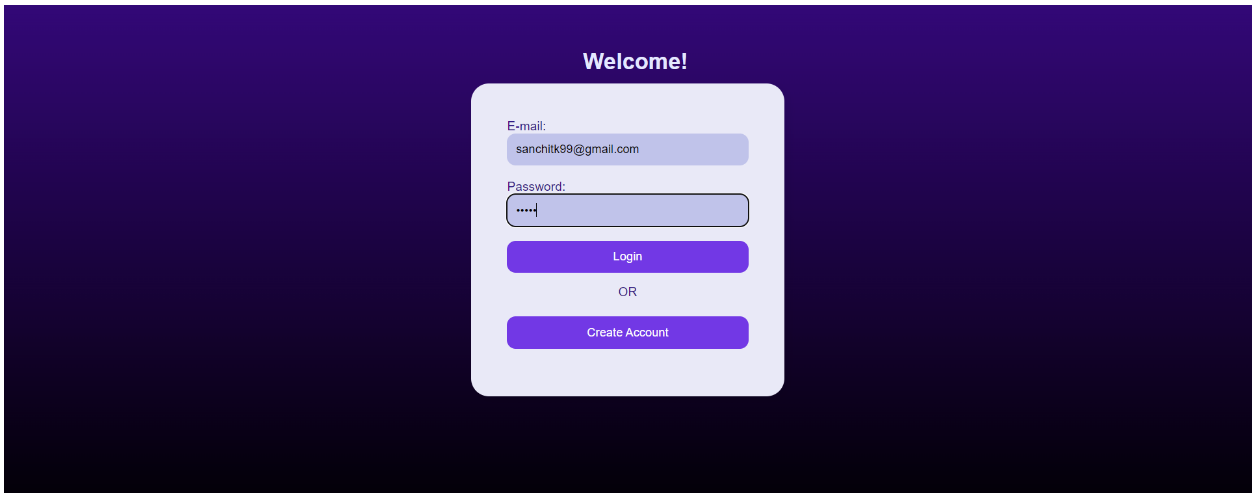
Address: 52 Rainbow Road

E-mail: allianad@gmail.com

TheatreID: 1

Employee Account Information Page: Lastly, we have the account information for the signed in employee. Here they can view the first and last name, birth date, address, email and theatre id that is saved under their account.

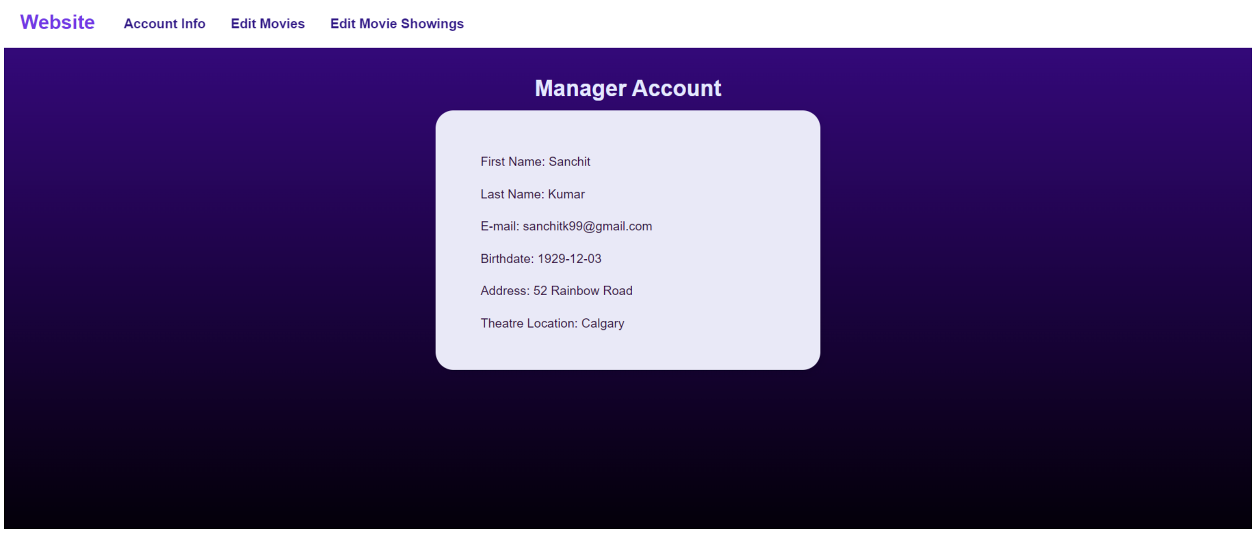
Manager Functionality:


Welcome!

E-mail: sanchitk99@gmail.com

Password: •••••

Login Page: First you must login to an existing manager account in the database (manager login shares the same login portal as employees). A new manager account cannot be created.



Manager Account

First Name: Sanchit

Last Name: Kumar

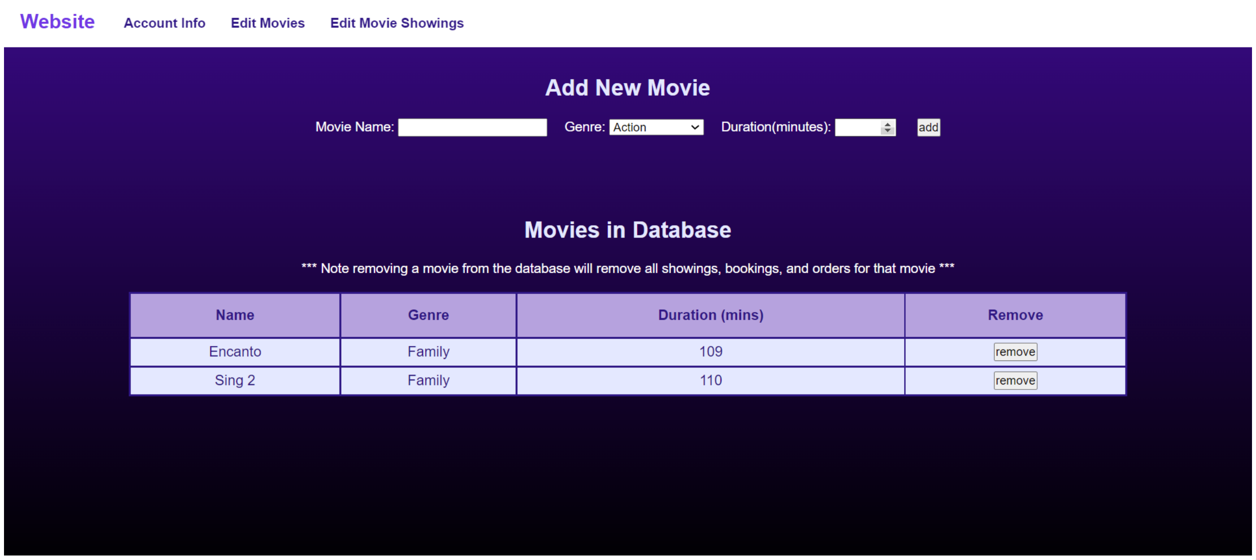
E-mail: sanchitk99@gmail.com

Birthdate: 1929-12-03

Address: 52 Rainbow Road

Theatre Location: Calgary

Account Info Page: After successfully signing in you are brought to the Account info page, which shows an account summary.



Add New Movie

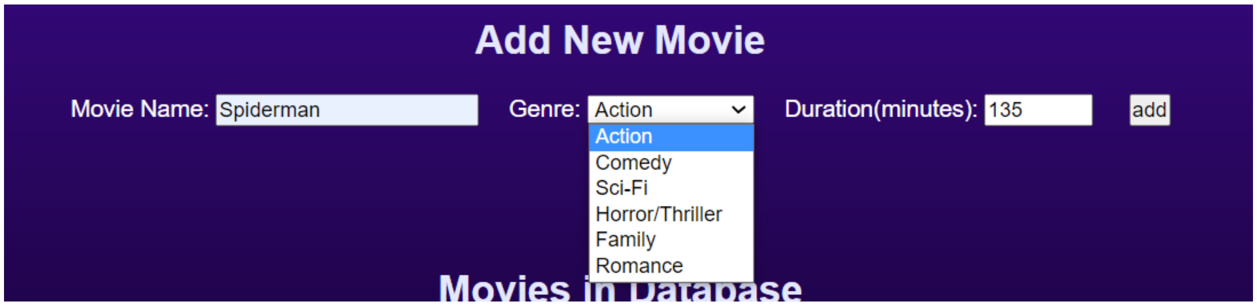
Movie Name: Genre: Action Duration(minutes): add

Movies in Database

*** Note removing a movie from the database will remove all showings, bookings, and orders for that movie ***

Name	Genre	Duration (mins)	Remove
Encanto	Family	109	remove
Sing 2	Family	110	remove

Edit Movies Page: The next tab is “Edit Movies”, where you can add/remove/view movies in the database.

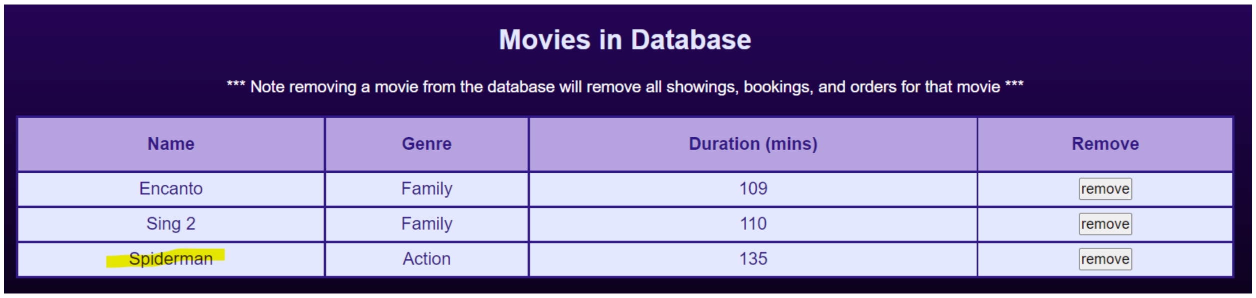


Add New Movie

Movie Name: Spiderman Genre: Action Duration(minutes): 135 add

Movies in Database

Edit Movies Page: Here managers can add new movies to the database by typing out the name, selecting a genre from the dropdown, and inserting its duration in minutes then pressing the “add” button.

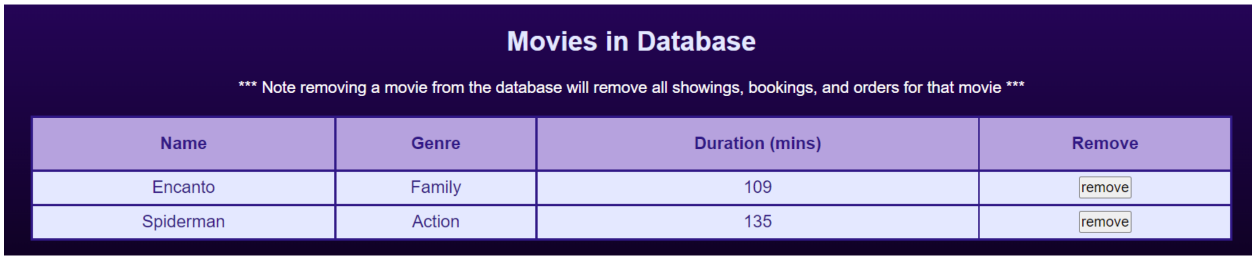


Movies in Database

*** Note removing a movie from the database will remove all showings, bookings, and orders for that movie ***

Name	Genre	Duration (mins)	Remove
Encanto	Family	109	remove
Sing 2	Family	110	remove
Spiderman	Action	135	remove

Edit Movies Page: After, a new movie should show up in the table below. (The movie “Spiderman” was added).

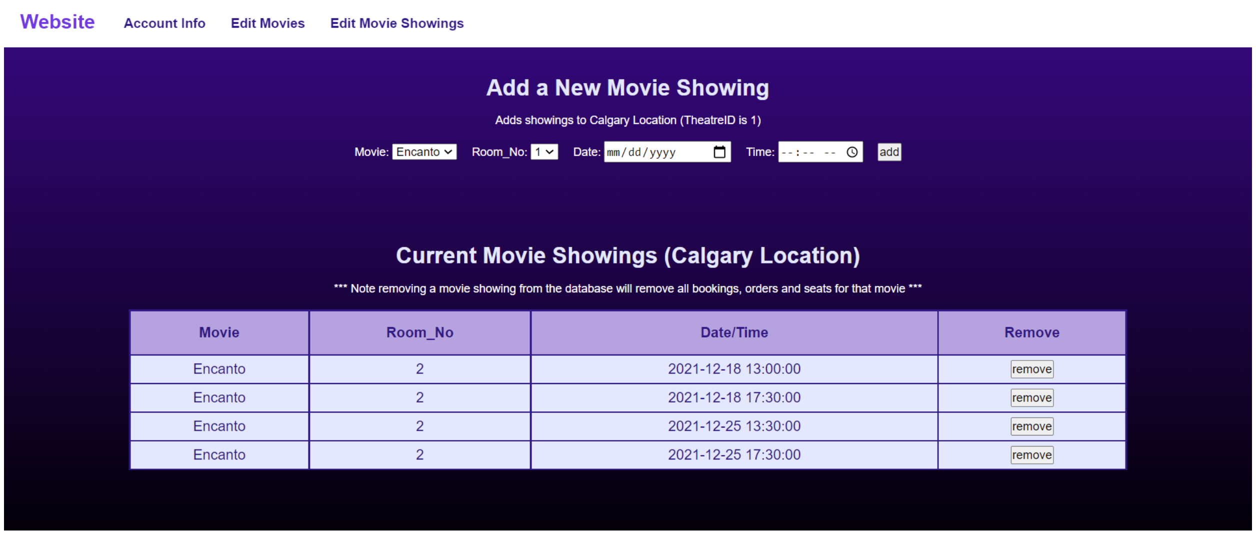


Movies in Database

*** Note removing a movie from the database will remove all showings, bookings, and orders for that movie ***

Name	Genre	Duration (mins)	Remove
Encanto	Family	109	remove
Spiderman	Action	135	remove

Edit Movies Page: Manager is also able to remove movies from the database by pressing the “remove” button to the right of the movie. (Above “Sing 2” was removed).



Add a New Movie Showing

Adds showings to Calgary Location (TheatreID is 1)

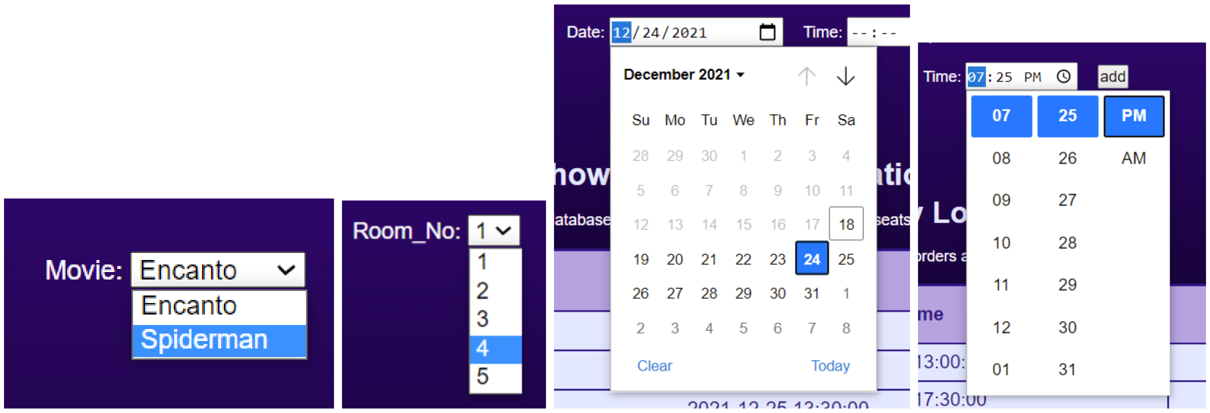
Movie: Encanto Room_No: 1 Date: mm/dd/yyyy Time: --:-- -- add

Current Movie Showings (Calgary Location)

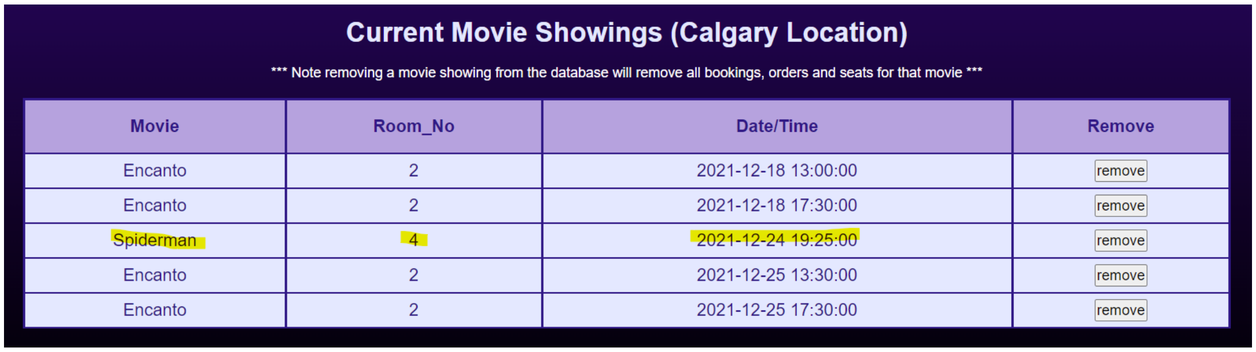
*** Note removing a movie showing from the database will remove all bookings, orders and seats for that movie ***

Movie	Room_No	Date/Time	Remove
Encanto	2	2021-12-18 13:00:00	remove
Encanto	2	2021-12-18 17:30:00	remove
Encanto	2	2021-12-25 13:30:00	remove
Encanto	2	2021-12-25 17:30:00	remove

Edit Movies Showings Page: In the next tab is “Edit Movie Showings”, here managers can add/remove/view movie showings for the theatre they manage.



Edit Movies Showings Page: When adding a new showing user must select a movie (that exists in the database), room number, select a date and start time, then press the “add” button.

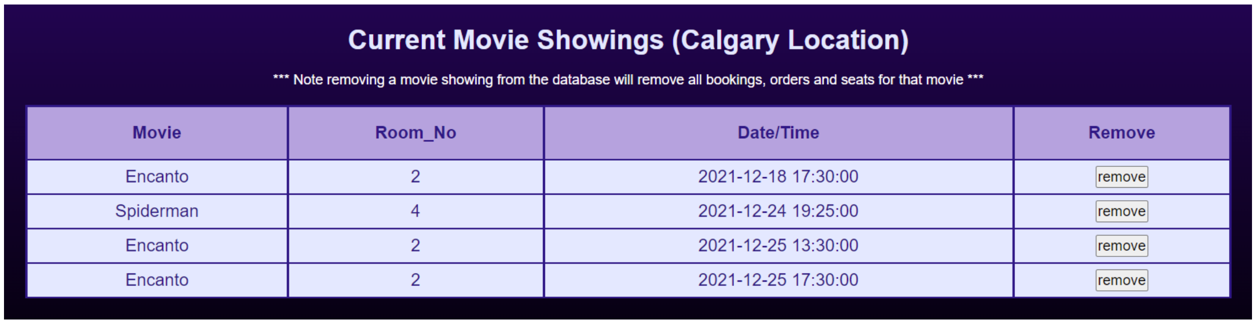


Current Movie Showings (Calgary Location)

*** Note removing a movie showing from the database will remove all bookings, orders and seats for that movie ***

Movie	Room_No	Date/Time	Remove
Encanto	2	2021-12-18 13:00:00	remove
Encanto	2	2021-12-18 17:30:00	remove
Spiderman	4	2021-12-24 19:25:00	remove
Encanto	2	2021-12-25 13:30:00	remove
Encanto	2	2021-12-25 17:30:00	remove

Edit Movies Showings Page: After, the new movie showing should be added to the table below. (“Spiderman” showing added in Room_No 4 on 2021-12-24 19:25:00). Note: The movie will not be added if it's time slot will conflict with other existing showings’ time slot.



Current Movie Showings (Calgary Location)

*** Note removing a movie showing from the database will remove all bookings, orders and seats for that movie ***

Movie	Room_No	Date/Time	Remove
Encanto	2	2021-12-18 17:30:00	remove
Spiderman	4	2021-12-24 19:25:00	remove
Encanto	2	2021-12-25 13:30:00	remove
Encanto	2	2021-12-25 17:30:00	remove

Edit Movies Showings Page: Manager is also able to remove the movie showing by using the “remove” button to the right of the showing. (Table after removing “Encanto” showing in Room_No 2 on 2021-12-18 at 17:30:00).

References

https://www.dailycal.org/2020/12/05/behind-the-curtains-a-brief-history-of-movie-theaters/
https://scripts.guru/add-minutes-to-date-and-time-in-php/