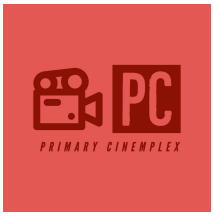

Movie Tickets Reservation Systems Project

Submitted to

Professor Thomas B. Capaul

Professor

TCSS 455 Database Systems Design

Submitted by

Kannika Armstrong, Nordine Chaumath, and Jessie De Jesus

UNIVERSITY of WASHINGTON

School of Engineering and Technology

University of Washington Tacoma

June 2022

Project Report

This report is phase III of the "Movies Tickets Reservation Systems Project" for the Database Systems Design course project in Fall 2022 at The University of Washington Tacoma. Phase III works on creating the conceptual model of the database and developing a functional front end of the database application to demonstrate the significant functionalities. The objectives of our project are

1. An online movie ticket booking system facilitates booking movie tickets for customers/members.
2. The E-ticketing system allows customers to browse through movies currently playing and book seats anywhere and anytime.
3. The movie ticket booking system is changing dynamically.

We started to work on project phase I with the discussion on the topic, the audience of our application, logos design, front-end design, and tools that will need to work on this project. First, we agreed on the discussion to create the database for the movie ticket reservation. Then, we decided to make the webpage front-end interface using -- PHP, HTML, Bootstraps, XAMPP, and phpMyAdmin.

In phase II of the project, we started to design the relational schema needed for the system and application. Since the audience of our application are customers/members, guests, and admins, we designed to have ten tables in the database. We tried to create the database in the Boyce-Codd Normal Form (BCNF) as much as possible. First, we wrote the SQL Data Definition Language (DDL) statements to create the database schema and the sample data set for our database (Appendix D). Then, we created the relational schemas diagram and wrote ten queries to test that our database works properly.

In the final phase, first, we created the Entity-Relationship (ER) Diagram of our database (Appendix A); then, we did the normalization of our relation schema. Second, we revised our database schema by splitting the 'users' table into the 'admin' table and 'customer' table because we thought those two audiences had different information to store in the database, as shown in the Relational Schema Diagram (Appendix C). Then, we do normalization again to ensure that our new relation schema is still in the BCNF (Appendix B). Once we were sure that our database was in the BCNF already. We started to design the front-end interface.

Before we started working on the front-end interface, we discussed the application features needed for our audience. So, our features will depend on the audience as follow:

1. All members can view movie schedules, book, and cancel tickets.
2. All guests can see the 'Now Playing' movie list, but they cannot book the ticket without logging in.
3. Admin has responsibility for adding new movie data, managing movie showtimes, and canceling any movie or show.

Finally, we decided to work on the front-end interface for the end-users, the customers/members, and the guests.

The features in our users' interface webpage are (Appendix E):

1. The first page was designed for every guest. Everyone can open that page to see our showtime playlist ('Now Playing') but cannot book any tickets without logging in or registering. We wrote the query to connect the page with the movie schema in the database. (Figure 10 - 12)
2. The registration page was designed for the new member to register to our website. We wrote the query to insert the new customer's information into the customer page. (Figure 13 - 17)
3. The login page was designed for the customer in our database can log in to the webpage to book the tickets. We wrote the query to connect the page with the customer schema in the database. (Figures 18 - 20)
4. The movie information page was designed to show each movie info from our database; title, poster, video, duration, rate genre, and movie info. (Figures 12 and 21)
5. The showtime schedule page was designed to show all schedules of each movie, including the location (branch name), date, start time, and end time. We wrote the query to connect the page with the showtime schema in the database. (Figures 22 - 24)
6. The seating page was designed to show all seats with both statuses, available and booked. We wrote the query to connect the page with the seat schema in the database to show the table status. And we also wrote the query to insert the new booked information into the booked schema with pending ('p') status, then

updated the seat status from available to booked. Finally, we wrote the query to insert the new ticket into the ticket schema. (Figures 25 - 32)

7. Finally, the user dashboard page was designed to show the pending ticket of each customer. We wrote the query to connect the page with several database schemas to show all the information needed for the dashboard page. (Figures 32 - 33)

While creating the webpage, we still need to revise our database -- adding more sample data to show on the webpage and adding more attributes needed for the real data system.
The challenges of this project are

1. Designing an efficient database.
2. Designing the front-end/user interface.
3. Normalizing the schemas.

This project is the starting project for us to work with the database and also gives us the ideas to expand the project or create new projects in the future. For example,

1. Re-design some database schemas.
2. Create the admin interface.
3. Create more schemas for the purchasing feature.
4. Design more schemas and interface for the front desk officer.
5. Design more schemas for the system features. For example, the main system is responsible for sending notifications for new movies, bookings, and cancellations.

We learned how to design an efficient database for the movie booking systems. We found that it is easier for the front-end developer to create the user interface if we have an efficient database. And it is also easy to modify when we need to add more schema or attributes, update the database, and store the data in the right place (schema). Furthermore, all schemas in the database related to each other somehow make it easy when we want to get some data/results from our database. Finally, a good database will always give us the correct output.

Appendix A: ER Diagram

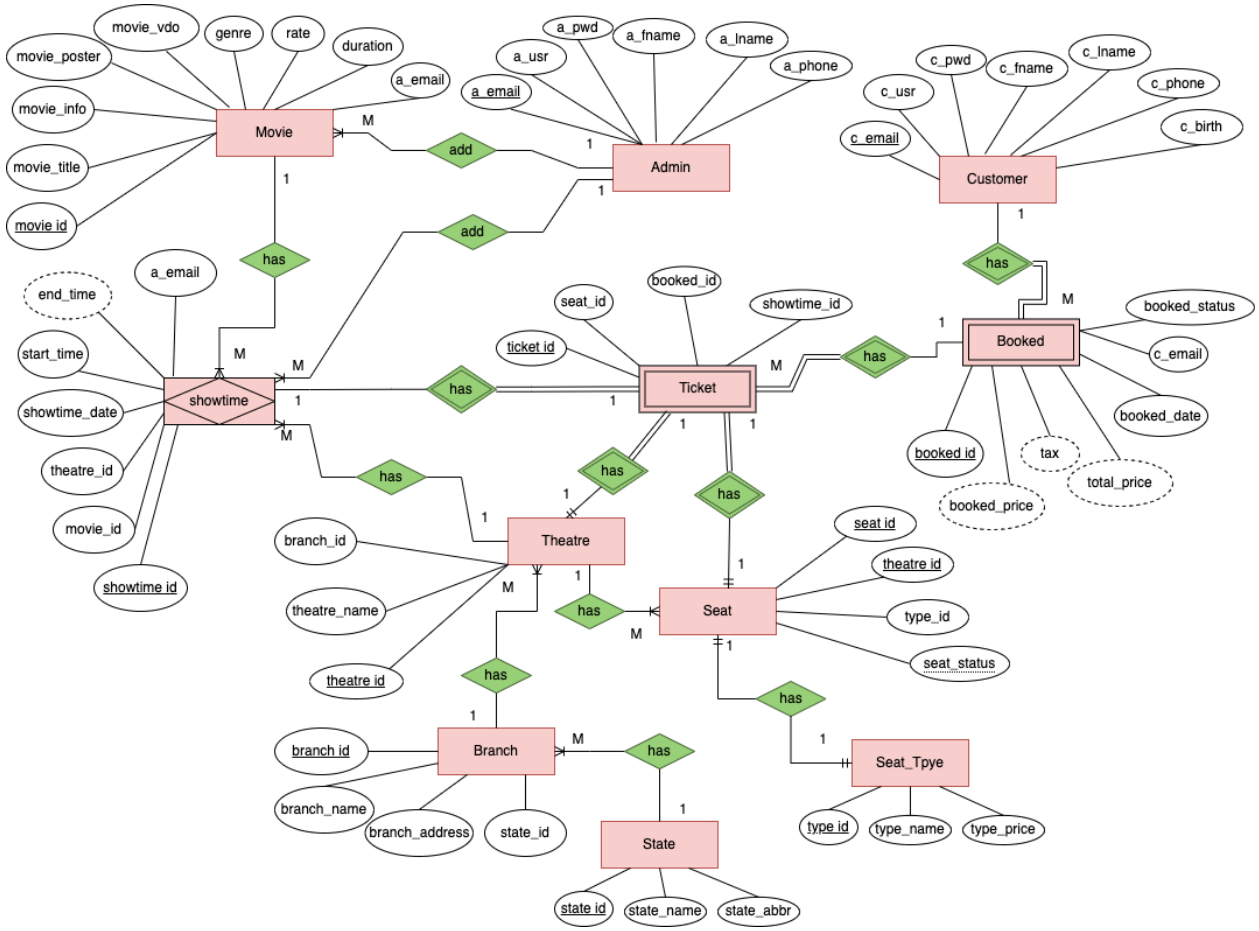

Figure 1 ER-diagram of the "Movies Tickets Reservation Systems Project"

Appendix B: Normalization Proof to BCNF

Admin:

a_email	a_usr	a_pwd	a_fname	a_lname	a_phone

FD: a_email → {a_usr, a_pwd, a_fname, a_lname, a_phone}

The attribute closure: a_email^+ = {a_email, a_usr, a_pwd, a_fname, a_lname, a_phone}

Proof: Since a_email can include all attributes of Admin, so a_email is a super key and also is a candidate key. ***This relation is in BCNF.***

Customer:

c_email	c_usr	c_pwd	c_fname	c_lname	c_birth	c_phone

FD: c_email → {c_usr, c_pwd, c_fname, c_lname, c_birth, c_phone}

The attribute closure: c_email^+ = {c_email, c_usr, c_pwd, c_fname, c_lname, c_birth, c_phone}

Proof: Since c_email can include all attributes of Customer, so c_email is a super key and also is a candidate key. ***This relation is in BCNF.***

Movie:

movie_id	movie_title	movie_info	movie_poster	movie_vdo	genre	rate	duration	a_email

FD: movie_id → {movie_title, movie_info, movie_poster, movie_vdo, genre, rate, duration, a_email}

The attribute closure: $movie\_id^+$ = {movie_id, movie_title, movie_info, movie_poster, movie_vdo, genre, rate, duration, a_email}

Proof: Since movie_id can include all attributes of Movie, so movie_id is a super key and also is a candidate key. ***This relation is in BCNF.***

Showtime:

showtime_id	movie_id	theatre_id	showtime_date	start_time	end_time	a_email

FD: showtime_id → {movie_id, theatre_id, showtime_date, start_time, end_time, a_email}

The attribute closure: showtime_id$^+$ = {showtime_id, movie_id, theatre_id, showtime_date, start_time, end_time, a_email}

Proof: Since showtime_id can includes all attributes of Showtime, so showtime_id is a super key and also is a candidate key. ***This relation is in BCNF.***

Theatre:

theatre_id	theatre_name	branch_id

FD: theatre_id → {theatre_name, branch_id}

The attribute closure: theatre_id$^+$ = {theatre_id, theatre_name, branch_id}

Proof: Since theatre_id can include all attributes of Theatre, so theatre_id is a super key and also is a candidate key. ***This relation is in BCNF.***

Branch:

branch_id	branch_name	branch_address	state_id

FD: branch_id → {branch_name, branch_address, state_id}

The attribute closure: branch_id$^+$ = {branch_id, branch_name, branch_address, state_id}

Proof: Since branch_id can include all attributes of Branch, so branch_id is a super key and also is a candidate key. This relation is in BCNF.

State:

state_id	state_name	state_abbr

FD: state_id → {state_name, state_abbr}

The attribute closure: state_id$^+$ = {state_id, state_name, state_abbr}

Proof: Since state_id can include all attributes of State, so state_id is a super key and also is a candidate key. ***This relation is in BCNF.***

Seat:

seat_id	theatre_id	type_id	seat_status

FD: seat_id, theatre_id → {type_id, seat_status}

The attribute closure: (seat_id, theatre_id)$^+$ = {seat_id, type_id, theatre_id, seat_status}

Proof: Since (state_id, theatre_id) can include all attributes of Seat. And neither $(seat\_id)^+$ nor $(theatre)^+$ cannot determine all attributes of Seat. So (state_id, theatre_id) is a super key and also is a candidate key. ***This relation is in BCNF.***

Seat_type:

type_id	type_name	type_price

FD: type_id → {type_name, type_price}

The attribute closure: $type\_id^+$ = {type_id, type_name, type_price}

Proof: Since type_id can include all attributes of Seat_type, so type_id is a super key and also is a candidate key. ***This relation is in BCNF.***

Ticket:

ticket_id	seat_id	theatre_id	booked_id	showtime_id

FD: ticket_id → {seat_id, theatre_id, booked_id, showtime_id}

The attribute closure: $ticket\_id^+$ = {ticket_id, seat_id, theatre_id, booked_id, showtime_id}

Proof: Since ticket_id can include all attributes of Ticket, so ticket_id is a super key and also is a candidate key. ***This relation is in BCNF.***

Booked:

booked_id	booked_price	tax	total_price	booked_date	c_email	booked_status

FD: booked_id → {booked_price, tax, total_price, booked_date, c_email, booked_status}

The attribute closure: $booked\_id^+$ = {booked_id, booked_price, tax, total_price, booked_date, c_email, booked_status}

Proof: Since booked_id can include all attributes of Booked, so booked_id is a super key and also is a candidate key. ***This relation is in BCNF.***

Appendix C: Relational Schema Diagram

Phase 2 Relational Schema

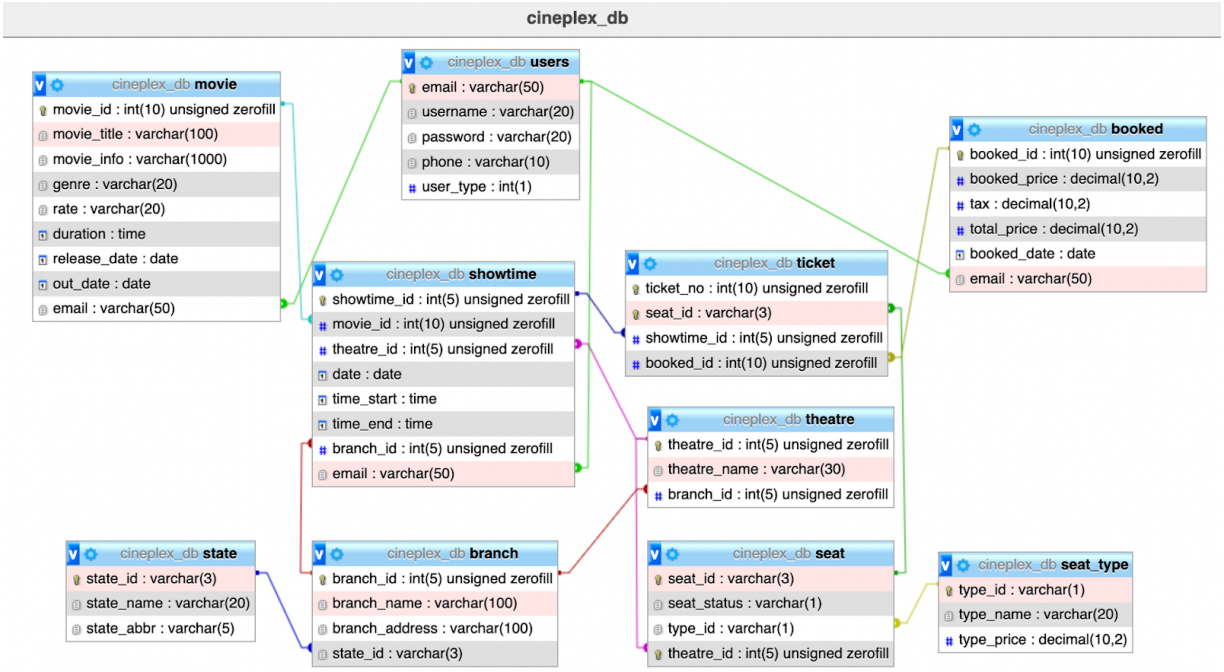

Figure 2 Relational Schema of the "Movies Tickets Reservation Systems Project" on phase 2

Phase 3 Relational Schema

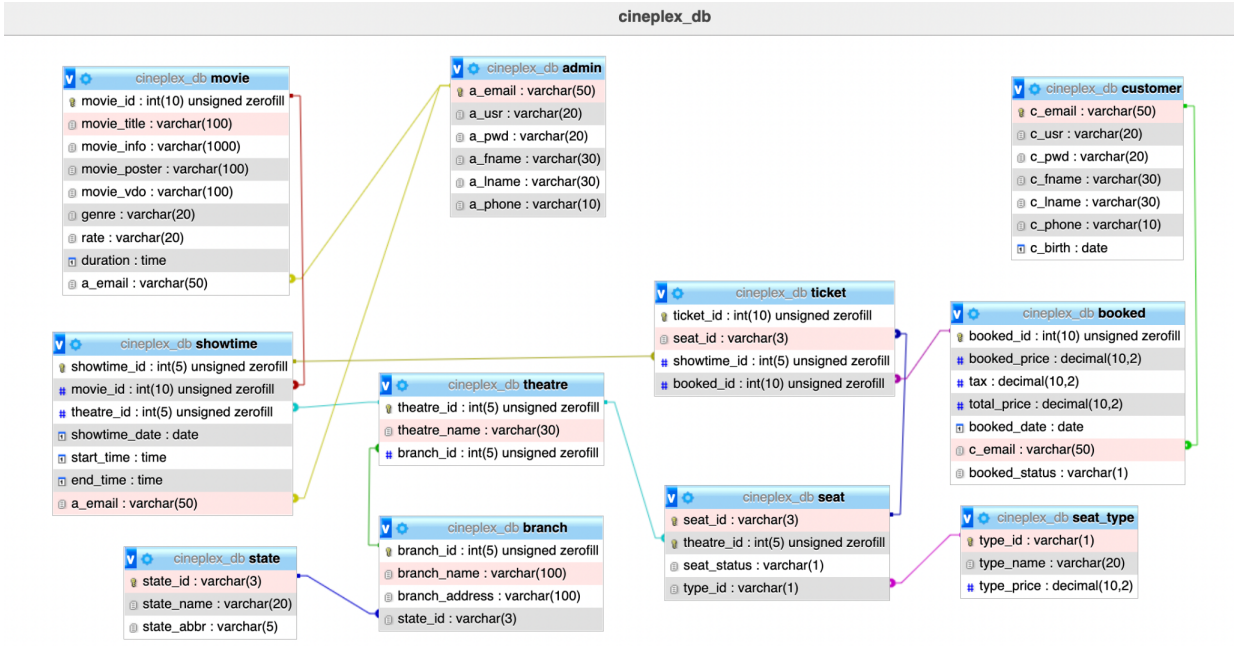

Figure 3 Relational Schema of the "Movies Tickets Reservation Systems Project" on phase 3

Appendix D: Screenshots of functional SQL Queries

In this appendix contains:

1. The screenshots of an example to create a movie table and booked table in our database. The picture shows that the movie relation schema and the booked relation schema are in the BCNF (Figure 4).
2. The screenshots of the state table and the branch table, which has the state_id from the state table as a foreign key with the constraint 'on delete cascade on update cascade' to ensure that they work properly.

```
CREATE TABLE if not exists `movie` (
  `movie_id` int(10) unsigned zerofill NOT NULL unique auto_increment,
  `movie_title` varchar(100) NOT NULL,
  `movie_info` varchar(1000) NOT NULL,
  `movie_poster` varchar(100) NOT NULL,
  `movie_vdo` varchar(100) NOT NULL,
  `genre` varchar(20) NOT NULL,
  `rate` varchar(20) NOT NULL,
  `duration` time NOT NULL,
  `a_email` varchar(50) NOT NULL default 'Admin',
   PRIMARY KEY (movie_id),
   FOREIGN KEY (a_email) REFERENCES admin (a_email)
       on delete restrict on update cascade
) ENGINE=InnoDB;

CREATE TABLE if not exists `booked` (
  `booked_id` int(10) unsigned zerofill NOT NULL unique auto_increment,
  `booked_price` decimal(10,2) NOT NULL,
  `tax` decimal(10,2) NOT NULL,
  `total_price` decimal(10,2) NOT NULL,
  `booked_date` date NOT NULL,
  `c_email` varchar(50) NOT NULL default 'Customer',
  `booked_status` varchar(1) NOT NULL,
   PRIMARY KEY (booked_id),
   FOREIGN KEY (c_email) REFERENCES customer (c_email)
       on delete restrict on update cascade
) ENGINE=InnoDB;
```

Figure 4 Example of functional SQL Queries to create movie table and booked table

```sql
CREATE TABLE if not exists `state` (
  `state_id` varchar(3) NOT NULL unique,
  `state_name` varchar(20) NOT NULL,
  `state_abbr` varchar(5) NOT NULL,
  PRIMARY KEY (state_id)
) ENGINE=InnoDB;
```

```sql
CREATE TABLE if not exists `branch` (
  `branch_id` int(5) unsigned zerofill NOT NULL unique auto_increment,
  `branch_name` varchar(100) NOT NULL,
  `branch_address` varchar(100) NOT NULL,
  `state_id` varchar(3) NOT NULL,
   PRIMARY KEY (branch_id),
   FOREIGN KEY (state_id) REFERENCES state (state_id)
      on delete cascade on update cascade
) ENGINE=InnoDB;
```

Figure 5 Example of functional SQL Queries to create state table and branch table

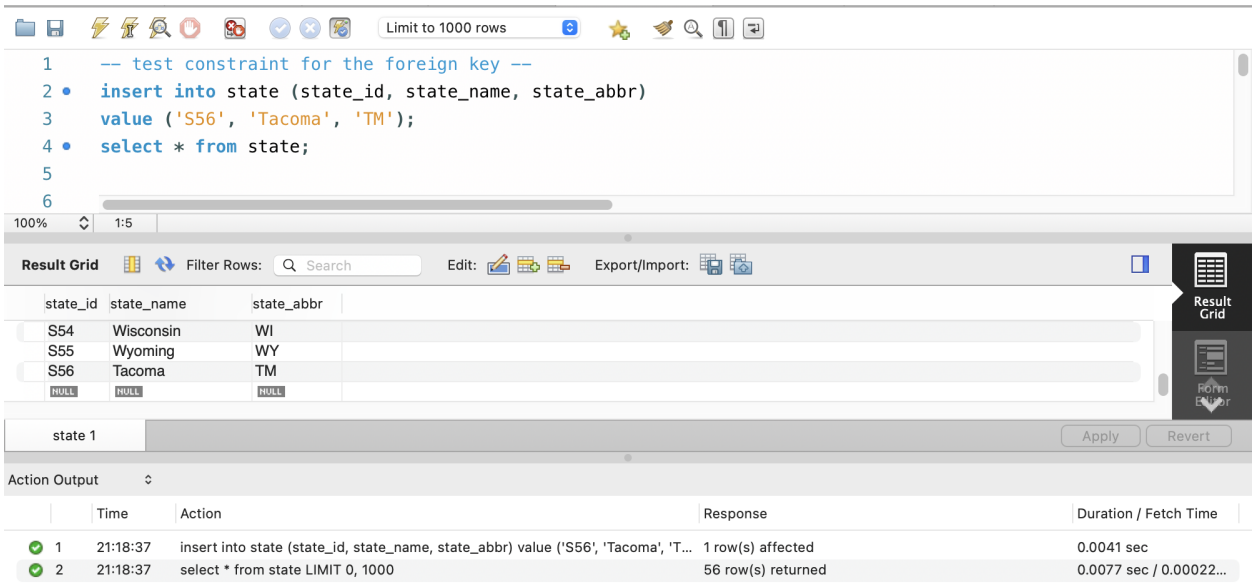

Figure 6 Insert new state in the state tale then insert the test branch in the new state

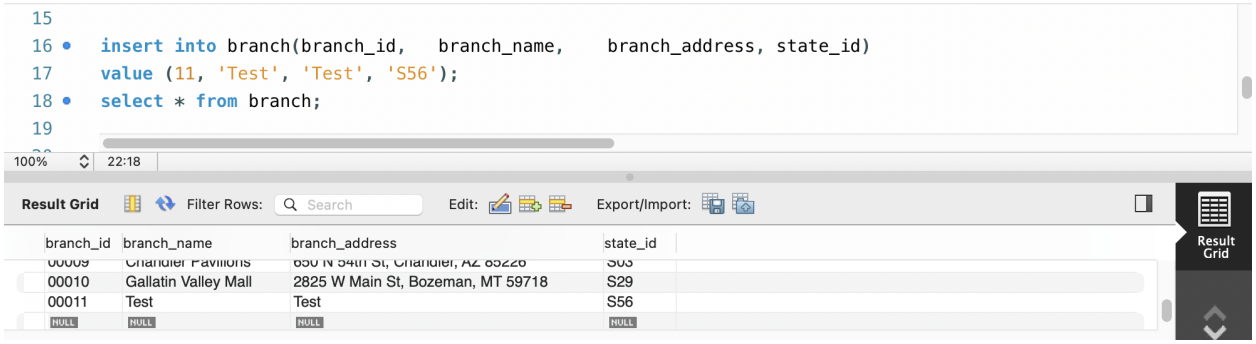

Figure 7 Insert the test branch in the new state to test the relationship between two table

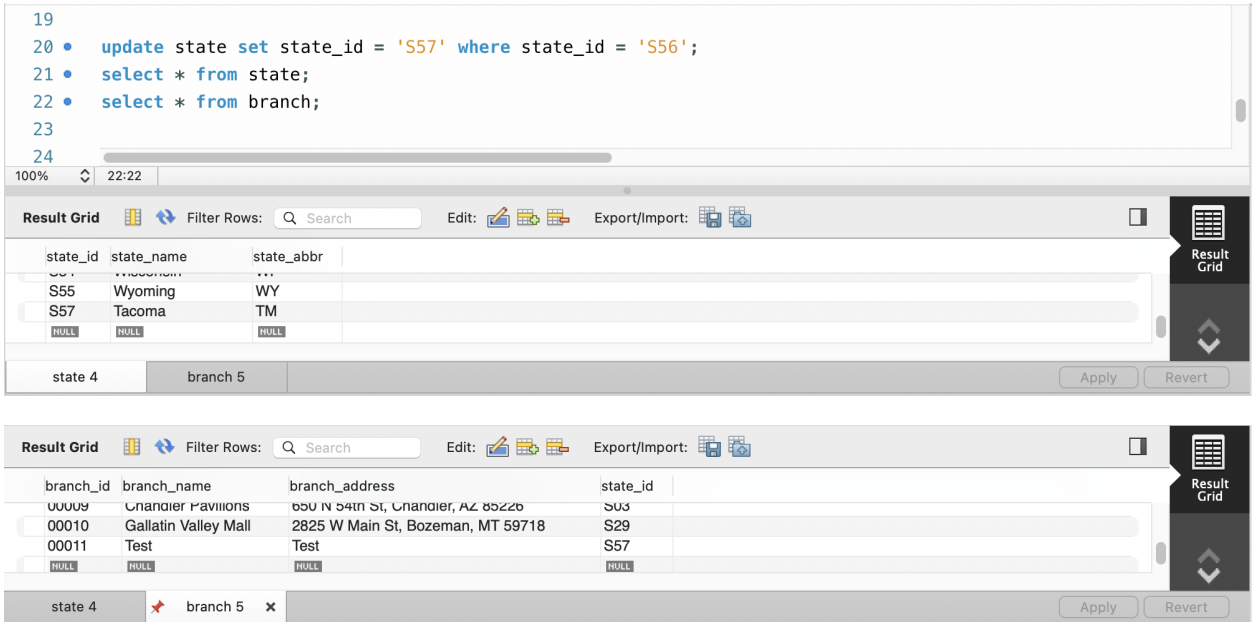

Figure 8 Test on update clause between two table

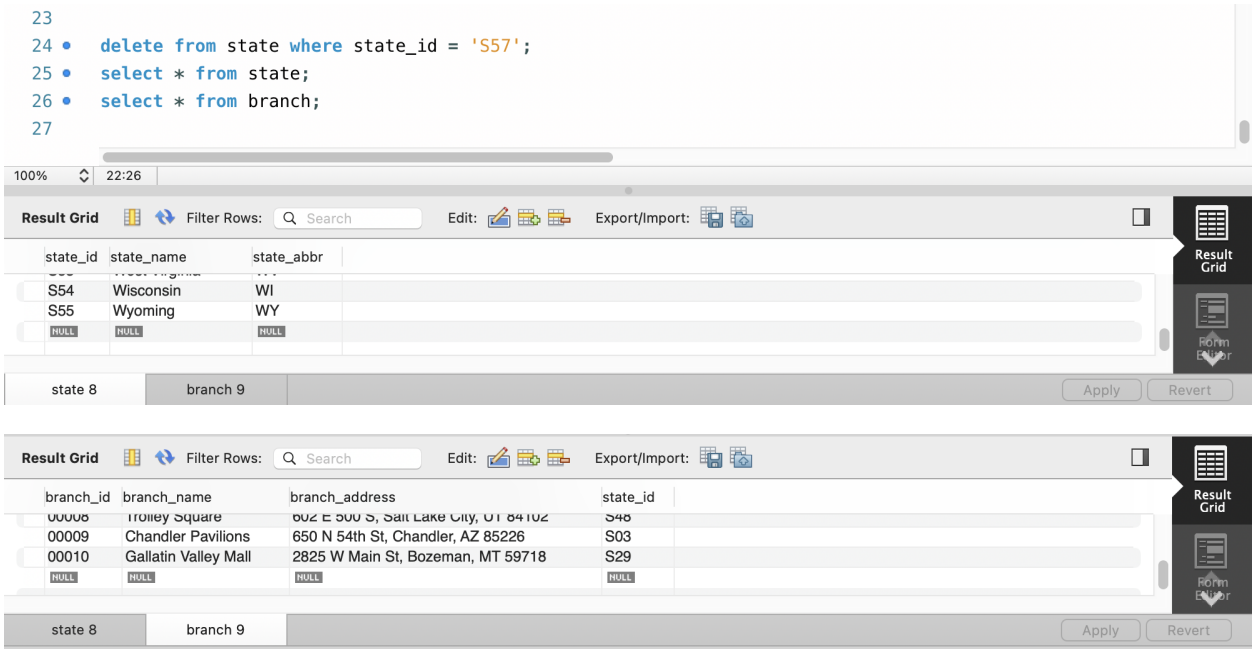

Figure 9 Test on delete clause between two tables

Appendix E: Screenshots of functional Web Interface

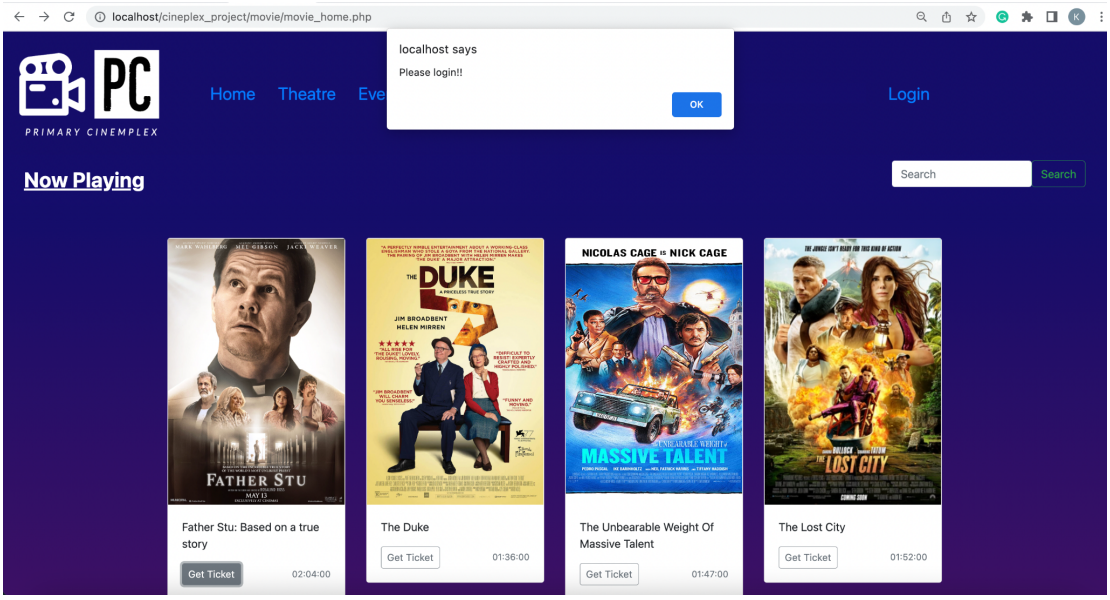

Figure 10 Screenshot of the first webpage that connects with the movie schema from the database

+ Options

	movie_id	movie_title	movie_info	movie_poster	movie_vdo	genre	rate	duration	a_email
Edit Copy Delete	0000000001	Doctor Strange In The Multiverse Of Madness	In Marvel Studios' "Doctor Strange in the Multiver...	img/pt01.jpeg	aWzlQ2N6qqg&t=7s	ACTION	PG13	02:06:00	admin1@cineplex.com
Edit Copy Delete	0000000002	The Bad Guys	Based on the bestselling book series by author Aar...	img/pt02.jpeg	m8Xt0yXaDPU	ANIMATION	PG	01:40:00	admin1@cineplex.com
Edit Copy Delete	0000000003	Sonic The Hedgehog 2	The world's favorite blue hedgehog is back for a n...	img/pt03.jpeg	G5kzUpWAusI	ANIMATION	PG	02:02:00	admin2@cineplex.com
Edit Copy Delete	0000000004	Everything Everywhere All at Once	When an interdimensional rupture unravels reality,...	img/pt04.jpeg	K2W_90S2NyM	SCIENCE FICTION	R	02:20:00	admin3@cineplex.com
Edit Copy Delete	0000000005	Fantastic Beasts: The Secrets of Dumbledore	Professor Albus Dumbledore (Law) knows the powerfu...	img/pt05.jpeg	Y9dr2zw-TXQ	ADVENTURE	PG13	02:22:00	admin1@cineplex.com
Edit Copy Delete	0000000006	The Northman	From visionary director Robert Eggers comes THE NO...	img/pt06.jpeg	oMSdFM12hOw	ACTION	R	02:17:00	admin3@cineplex.com
Edit Copy Delete	0000000007	The Lost City	Brilliant, but reclusive author Loretta Sage (Sand...	img/pt07.jpeg	nfKO9rYDmE8	COMEDY	PG13	01:52:00	admin2@cineplex.com
Edit Copy Delete	0000000008	The Unbearable Weight Of Massive Talent	Nicolas Cage stars as... Nick Cage in the action-c...	img/pt08.jpeg	x2YHPZMj8r4	COMEDY	R	01:47:00	admin1@cineplex.com
Edit Copy Delete	0000000009	The Duke	THE DUKE is set in 1961 when Kempton Bunton, a 60-...	img/pt09.jpeg	R4PkA26wEA0	COMEDY	R	01:36:00	admin1@cineplex.com
Edit Copy Delete	0000000010	Father Stu: Based on a true story	Father Stu is an unflinchingly honest, funny and u...	img/pt10.jpeg	DHREzAdyCPs	DRAMA	R	02:04:00	admin3@cineplex.com

Figure 11 Screenshot of the movie schema from the database

```php
<?php

$query = mysqli_query($conn, 'SELECT * FROM movie ORDER BY movie_id DESC');

while($result = mysqli_fetch_array($query)) {

?>
        <div class="col-md-3">
        <div class="card mb-4 shadow-sm">
        <img src=<?=$result['movie_poster']?> width="100%" height="380" />
        <div class="card-body">

        <p class="card-text"><?=$result['movie_title']?></p>

        <div class="d-flex justify-content-between align-items-center">
                <div class="btn-group">
                <button type="button" class="btn btn-sm btn-outline-secondary">
                        <a href="./movies_page.php?index=<?=$result['movie_id']?>">Get Ticket</a></button>
                </div>
                <small class="text-muted"><?=$result['duration']?></small>
        </div>
        </div>
        </div>
        </div>
<?php }
?>
```

Figure 12 Screenshot of the SQL query connecting the database with the webpage

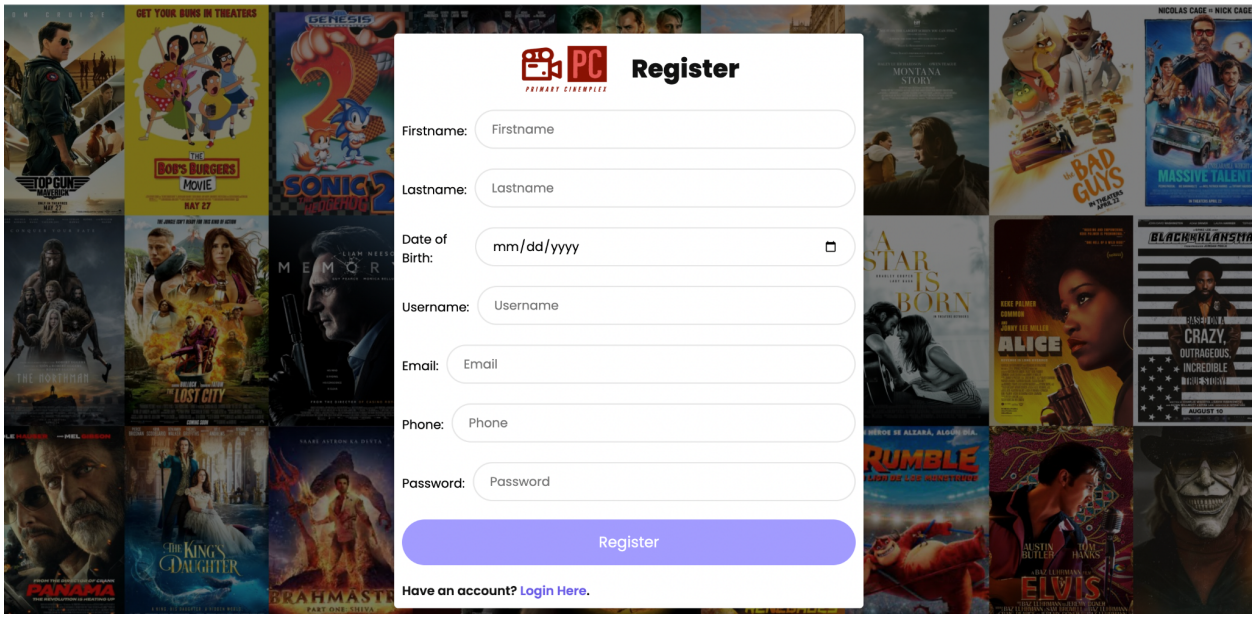

Figure 13 Screenshot of the registration webpage to insert new member's information into the customer table in the database

+ Options

	c_email	c_usr	c_pwd	c_fname	c_lname	c_phone	c_birth
Edit Copy Delete	customer1@gmail.com	customer1	IamACustomer1	Nook	Noi	011119999	1963-02-12
Edit Copy Delete	customer2@gmail.com	customer2	Customer2IsMe	Aroon	Kamol	012229999	2000-05-11
Edit Copy Delete	customer3@gmail.com	customer3	HelloAmCust3	Joy	Jones	013339999	1999-06-30
Edit Copy Delete	customer4@gmail.com	customer4	IamNo4	Amy	Army	014449999	1980-08-04
Edit Copy Delete	customer5@gmail.com	customer5	5IsHere	Booko	Boku	015559999	1984-06-08
Edit Copy Delete	customer6@gmail.com	customer6	LetsMeBe6	Chook	Kim	016669999	1960-10-10
Edit Copy Delete	customer7@gmail.com	customer7	ThisIsJust7	Kimmy	Lee	017779999	1978-12-01
Edit Copy Delete	customer8@gmail.com	customer8	WoohooAm8	Mark	Prince	018889999	1998-12-31
Edit Copy Delete	customer9@gmail.com	customer9	Before10	Pual	Suwat	019999999	1992-04-17

Figure 14 Screenshot of the customer table in the database before creating a new account

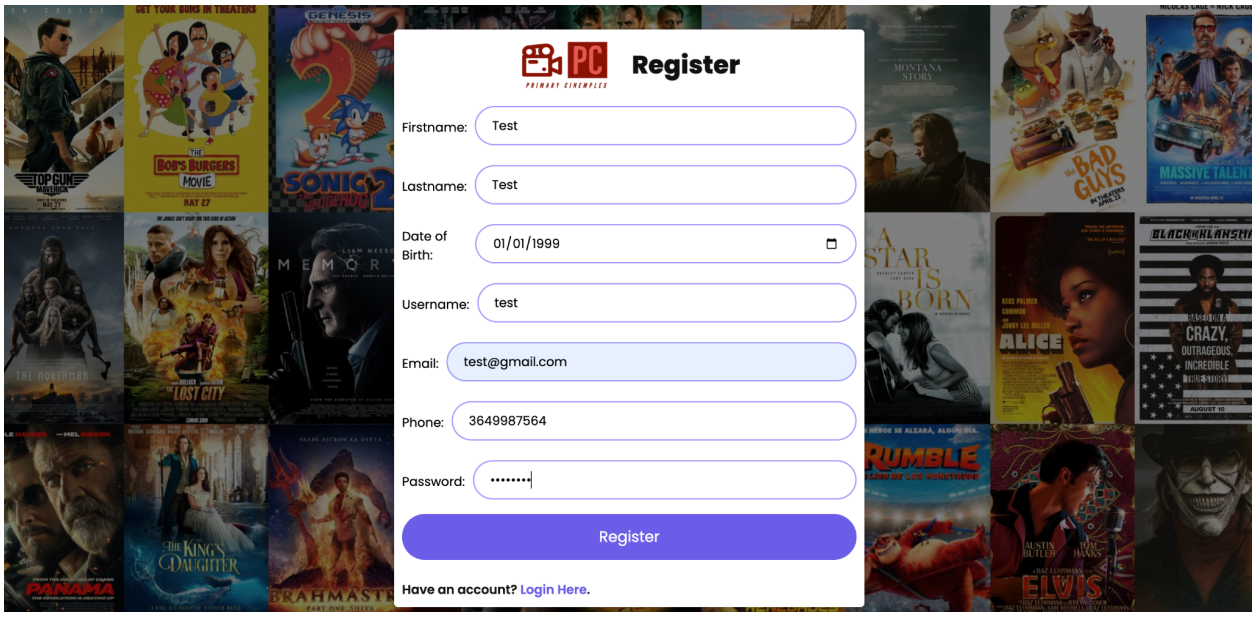

Figure 15 Screenshot of creating the new account from the registration page

+ Options

	c_email	c_usr	c_pwd	c_fname	c_lname	c_phone	c_birth
Edit Copy Delete	customer1@gmail.com	customer1	IamACustomer1	Nook	Noi	011119999	1963-02-12
Edit Copy Delete	customer2@gmail.com	customer2	Customer2IsMe	Aroon	Kamol	012229999	2000-05-11
Edit Copy Delete	customer3@gmail.com	customer3	HelloAmCust3	Joy	Jones	013339999	1999-06-30
Edit Copy Delete	customer4@gmail.com	customer4	IamNo4	Amy	Army	014449999	1980-08-04
Edit Copy Delete	customer5@gmail.com	customer5	5IsHere	Booko	Boku	015559999	1984-06-08
Edit Copy Delete	customer6@gmail.com	customer6	LetsMeBe6	Chook	Kim	016669999	1960-10-10
Edit Copy Delete	customer7@gmail.com	customer7	ThisIsJust7	Kimmy	Lee	017779999	1978-12-01
Edit Copy Delete	customer8@gmail.com	customer8	WoohooAm8	Mark	Prince	018889999	1998-12-31
Edit Copy Delete	customer9@gmail.com	customer9	Before10	Pual	Suwat	019999999	1992-04-17
Edit Copy Delete	test@gmail.com	test	password	Test	Test	3649987564	1999-01-01

Figure 16 Screenshot of the customer table in the database after creating a new account

```
if($conn->connect_error){
        die("Connection Failed:  ".$conn->connect_error);
} else {
        $stmt = $conn->prepare('INSERT INTO customer(c_email, c_usr, c_pwd, c_fname, c_lname, c_phone, c_birth)
                                VALUES(?, ?, ?, ?, ?, ?, ?)');
        $stmt->bind_param('sssssss', $email, $username, $password, $firstname, $lastname, $phone, $birth);
        $stmt->execute();
        echo '<script type="text/javascript">
        alert("Your account is ready!" + "\n" + "Please login.");
        window.location = "customer_login.php";
    </script>';
        $stmt->close();
        $conn->close();
}
?>
```

Figure 17 Screenshot of the SQL query to insert new customer's information into the customer table

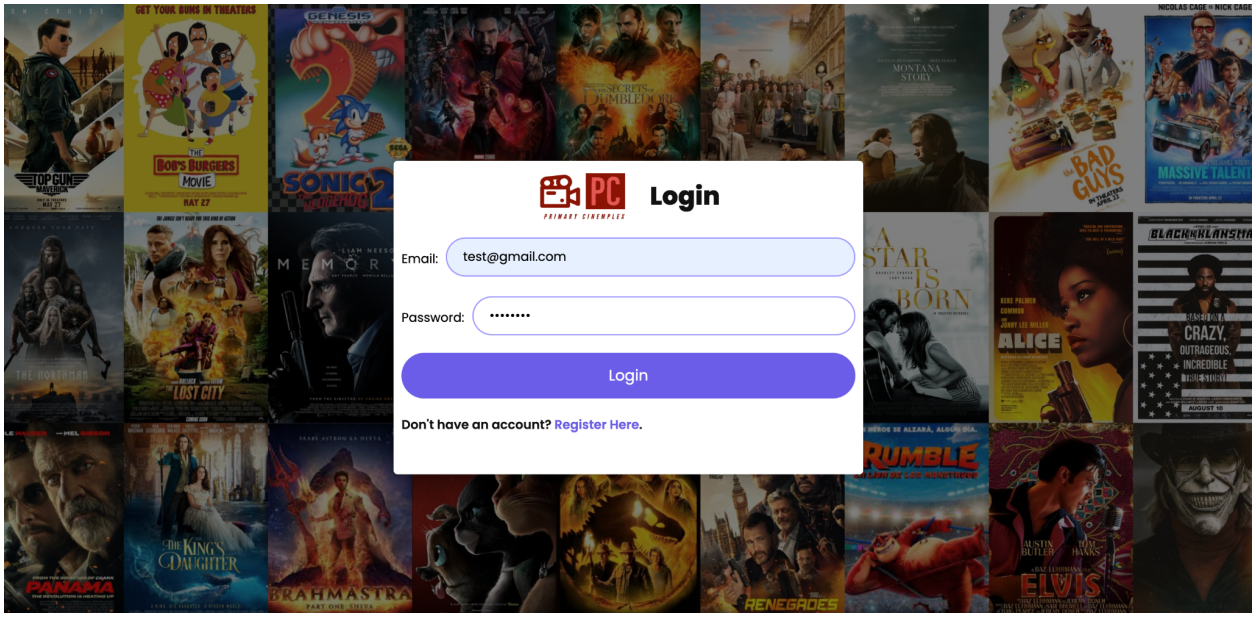

Figure 18 Screenshot of logging in to the webpage from the login page

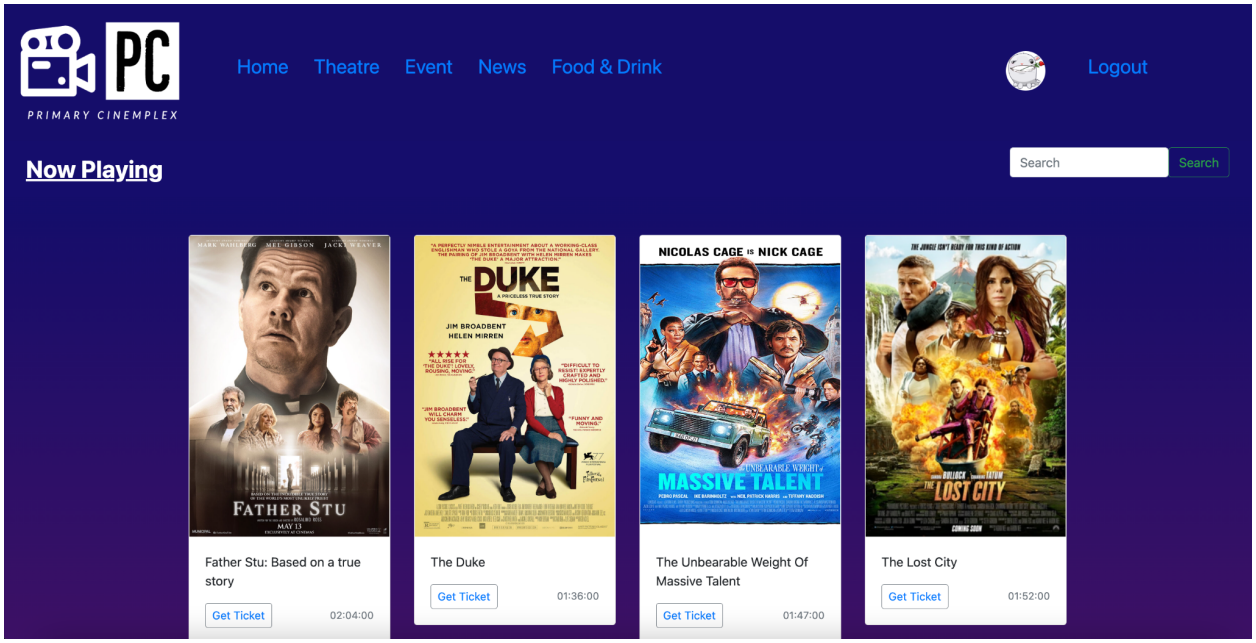

Figure 19 Screenshot of the first webpage after login the 'Get ticket' button is active

```php
$sql = "SELECT * FROM customer WHERE c_email='$uname' AND c_pwd='$pass'";

$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) === 1) {
        $row = mysqli_fetch_assoc($result);

        $_SESSION['c_email'] = $row['c_email'];
        $_SESSION['c_usr'] = $row['c_usr'];
        $_SESSION['c_pwd'] = $row['c_pwd'];
        $_SESSION['c_fname'] = $row['c_fname'];
        $_SESSION['c_lname'] = $row['c_lname'];
        $_SESSION['c_phone'] = $row['c_phone'];
        $_SESSION['c_birth'] = $row['c_birth'];

        if ($row['c_email'] === $uname && $row['c_pwd'] === $pass) {
                header('Location: ../movie/movie_index.php');
                exit();
        }else{
                header("Location: customer_index.php?error=Incorect Email or password");
                exit();
        }
}else{
        header("Location: customer_index.php?error=Incorect Email or password");
        exit();
}
```

Figure 20 Screenshot of the SQL query to login to the webpage with the customer's email from the database

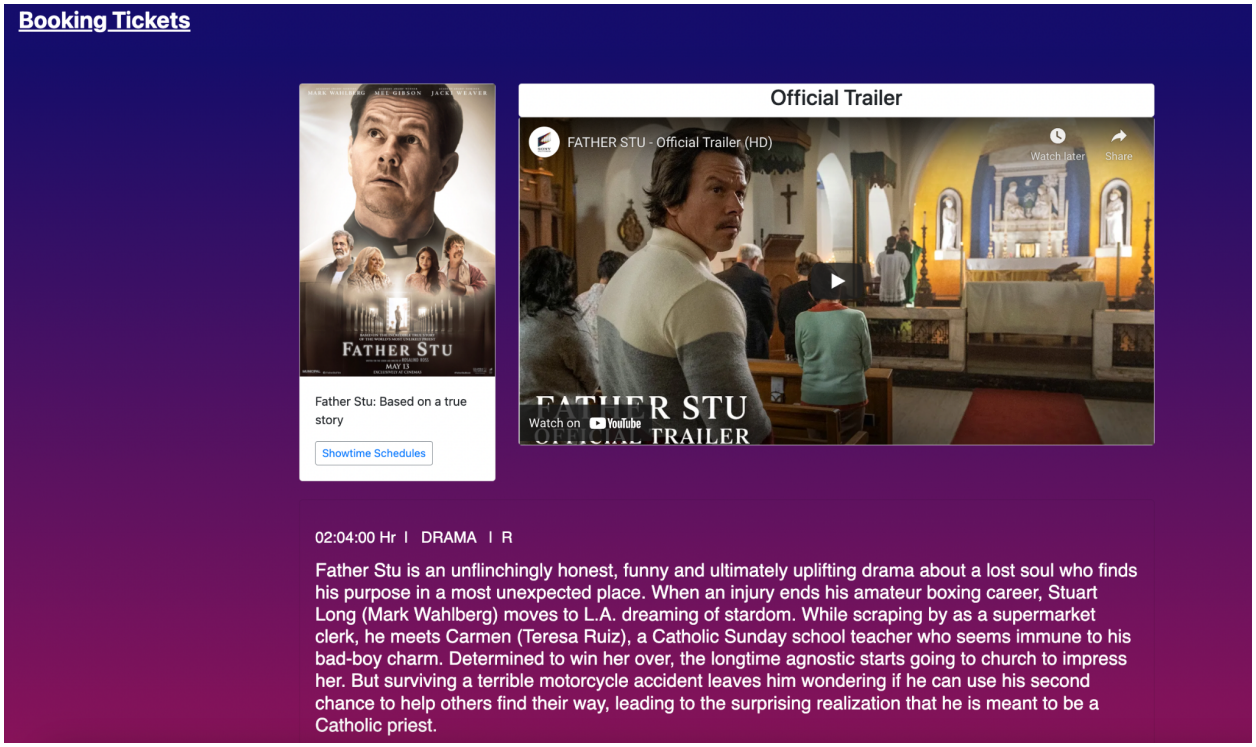

Figure 21 Screenshot of the webpage after clicking the 'Get ticket' button

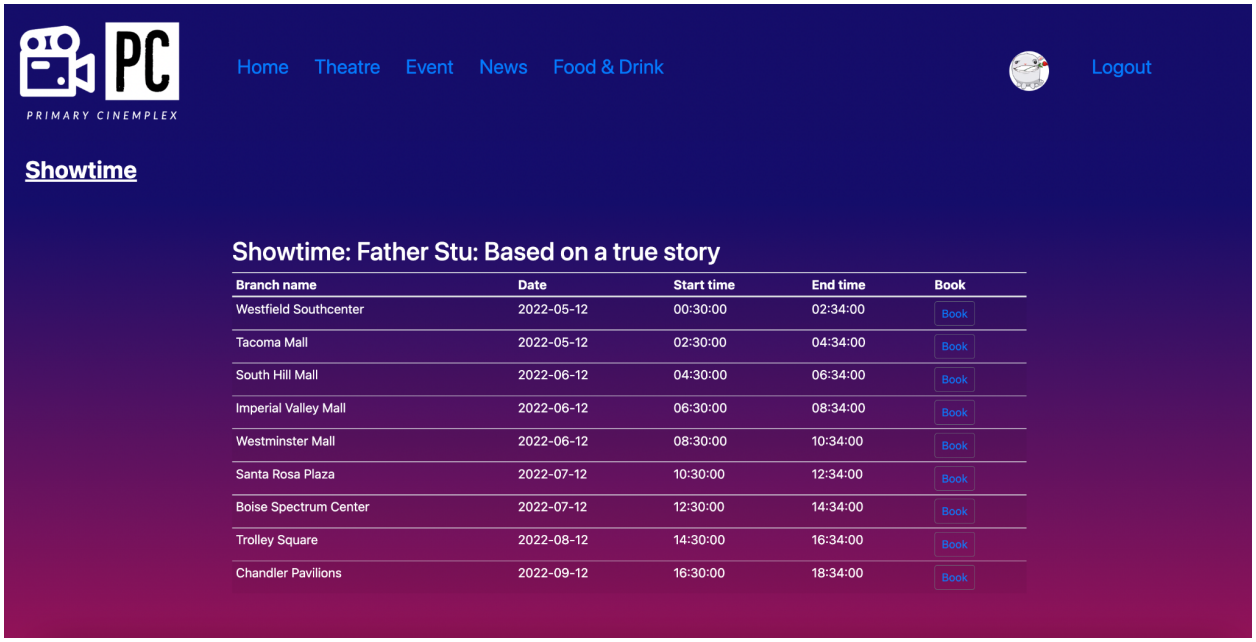

Figure 22 Screenshot of the webpage after clicking the 'Showtime Schedules' button

+ Options

	showtime_id	movie_id	theatre_id	showtime_date	start_time	end_time	a_email
Edit Copy Delete	00001	0000000001	00009	2022-05-11	08:30:00	10:36:00	admin1@cineplex.com
Edit Copy Delete	00002	0000000001	00001	2022-05-11	08:30:00	10:10:00	admin1@cineplex.com
Edit Copy Delete	00003	0000000002	00002	2022-05-12	08:30:00	10:32:00	admin3@cineplex.com
Edit Copy Delete	00004	0000000001	00001	2022-05-12	11:00:00	13:20:00	admin3@cineplex.com
Edit Copy Delete	00005	0000000006	00004	2022-05-11	11:00:00	13:22:00	admin2@cineplex.com
Edit Copy Delete	00006	0000000002	00004	2022-05-11	13:30:00	15:47:00	admin2@cineplex.com
Edit Copy Delete	00007	0000000001	00008	2022-05-12	13:30:00	15:22:00	admin1@cineplex.com
Edit Copy Delete	00008	0000000008	00009	2022-05-12	17:30:00	19:17:00	admin3@cineplex.com
Edit Copy Delete	00009	0000000001	00001	2022-05-12	10:30:00	22:06:00	admin2@cineplex.com
Edit Copy Delete	00010	0000000002	00010	2022-05-12	17:30:00	21:34:00	admin2@cineplex.com
Edit Copy Delete	00011	0000000010	00001	2022-05-12	00:30:00	02:34:00	admin1@cineplex.com
Edit Copy Delete	00012	0000000010	00004	2022-05-12	02:30:00	04:34:00	admin2@cineplex.com
Edit Copy Delete	00013	0000000010	00007	2022-06-12	04:30:00	06:34:00	admin3@cineplex.com
Edit Copy Delete	00014	0000000010	00010	2022-06-12	06:30:00	08:34:00	admin1@cineplex.com
Edit Copy Delete	00015	0000000010	00013	2022-06-12	08:30:00	10:34:00	admin2@cineplex.com
Edit Copy Delete	00016	0000000010	00016	2022-07-12	10:30:00	12:34:00	admin3@cineplex.com
Edit Copy Delete	00017	0000000010	00019	2022-07-12	12:30:00	14:34:00	admin1@cineplex.com
Edit Copy Delete	00018	0000000010	00022	2022-08-12	14:30:00	16:34:00	admin2@cineplex.com
Edit Copy Delete	00019	0000000010	00025	2022-09-12	16:30:00	18:34:00	admin3@cineplex.com

Figure 23 Screenshot of the showtime table in the database

```php
<?php

    $query = mysqli_query($conn, "SELECT * FROM showtime
    JOIN theatre USING (theatre_id)
    JOIN branch USING (branch_id)
    Where movie_id = $id
    ORDER BY showtime_date, start_time");

    while($result = mysqli_fetch_array($query)) {
        $_SESSSION[$result['branch_id']] = $result['branch_id'];

    ?>

        <tr>
                <td><?=$result['branch_name']?></td>
                <td><?=$result['showtime_date']?></td>
                <td><?=$result['start_time']?></td>
                <td><?=$result['end_time']?></td>
                <td><button type="button" class="btn btn-sm btn-outline-secondary"><a href="./movie_seating.php?index=<?=$result['movie_id']?>&branch=<?=$

        </tr>
<?php }
?>
```

Figure 24 Screenshot of the SQL query to show all showtime schedules of the movies

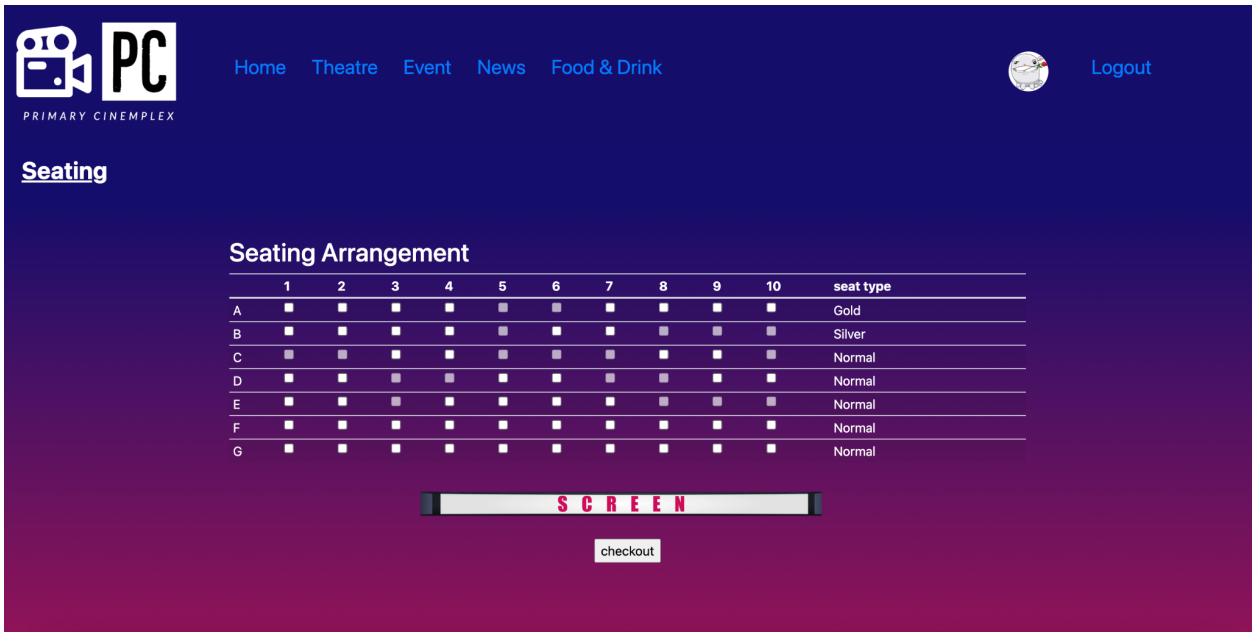

Figure 25 Screenshot of the webpage after clicking the 'Book' button on the showtime page

```php
<?php

            $query = mysqli_query($conn, "SELECT *
            FROM showtime JOIN theatre USING (theatre_id) JOIN branch USING (branch_id) JOIN seat USING (theatre_id)
            Where movie_id = $id AND branch_id = $branch AND theatre_id = $theatre AND seat_status = 'B'");

            while($result = mysqli_fetch_array($query)) {
                    $seat = $result['seat_id'];
                    echo "<script>
                            document.getElementById('$seat').disabled = true;
                            </script>";

            ?>

            <?php }
    ?>
```

Figure 26 Screenshot of the SQL query to show all available and booked seat of the show

+ Options

booked_id	booked_price	tax	total_price	booked_date	c_email	booked_status
0000000001	34.00	3.40	37.40	2022-05-11	customer1@gmail.com	c
0000000002	45.00	4.50	49.50	2022-05-11	customer2@gmail.com	c
0000000003	12.00	1.20	13.20	2022-05-11	customer3@gmail.com	c
0000000004	24.00	2.40	26.40	2022-05-11	customer4@gmail.com	c
0000000005	24.00	2.40	26.40	2022-05-11	customer5@gmail.com	c
0000000006	24.00	2.40	26.40	2022-05-11	customer6@gmail.com	c
0000000007	12.00	1.20	13.20	2022-05-11	customer7@gmail.com	c
0000000008	36.00	3.60	39.60	2022-05-11	customer8@gmail.com	c
0000000009	36.00	3.60	39.60	2022-05-11	customer8@gmail.com	c
0000000010	15.00	1.50	16.50	2022-05-11	customer5@gmail.com	c

Figure 27 Screenshot of the booked table in the database before booking the new ticket

+ Options

ticket_id	seat_id	showtime_id	booked_id
0000000001	A05	00004	0000000001
0000000002	A06	00004	0000000001
0000000003	B08	00004	0000000002
0000000004	B09	00004	0000000002
0000000005	B10	00004	0000000002
0000000006	E03	00004	0000000003
0000000007	D03	00004	0000000004
0000000008	D04	00004	0000000004
0000000009	D07	00004	0000000005
0000000010	D08	00004	0000000005
0000000011	C01	00004	0000000006
0000000012	C02	00004	0000000006
0000000013	C10	00004	0000000007
0000000014	E08	00004	0000000008
0000000015	E09	00004	0000000008
0000000016	E10	00004	0000000008
0000000017	C05	00004	0000000009
0000000018	C06	00004	0000000009
0000000019	C07	00004	0000000009
0000000020	B05	00004	0000000010

Figure 28 Screenshot of the ticket table in the database before booking the new ticket

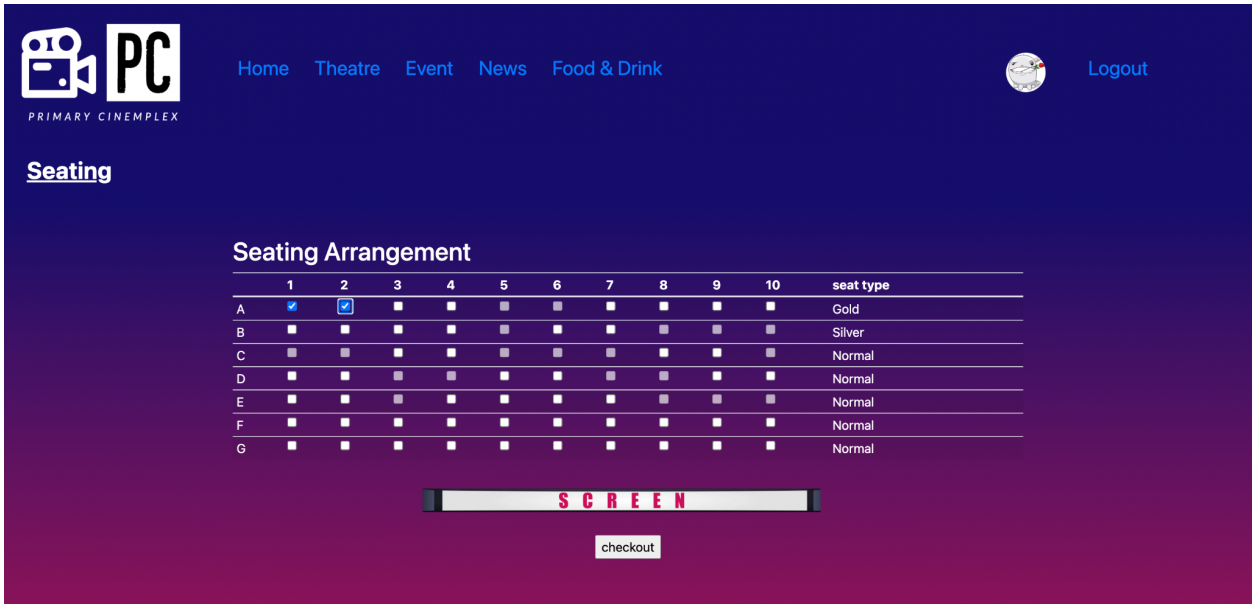

Figure 29 Screenshot of the webpage with selecting seat numbers A01 and A02

+ Options

	booked_id	booked_price	tax	total_price	booked_date	c_email	booked_status
Edit Copy Delete	0000000001	34.00	3.40	37.40	2022-05-11	customer1@gmail.com	c
Edit Copy Delete	0000000002	45.00	4.50	49.50	2022-05-11	customer2@gmail.com	c
Edit Copy Delete	0000000003	12.00	1.20	13.20	2022-05-11	customer3@gmail.com	c
Edit Copy Delete	0000000004	24.00	2.40	26.40	2022-05-11	customer4@gmail.com	c
Edit Copy Delete	0000000005	24.00	2.40	26.40	2022-05-11	customer5@gmail.com	c
Edit Copy Delete	0000000006	24.00	2.40	26.40	2022-05-11	customer6@gmail.com	c
Edit Copy Delete	0000000007	12.00	1.20	13.20	2022-05-11	customer7@gmail.com	c
Edit Copy Delete	0000000008	36.00	3.60	39.60	2022-05-11	customer8@gmail.com	c
Edit Copy Delete	0000000009	36.00	3.60	39.60	2022-05-11	customer8@gmail.com	c
Edit Copy Delete	0000000010	15.00	1.50	16.50	2022-05-11	customer5@gmail.com	c
Edit Copy Delete	0000000011	34.00	3.40	37.40	2022-06-10	test@gmail.com	p

Figure 30 Screenshot of the booked table in the database after booking the new ticket with the pending ('p') status

+ Options

	ticket_id	seat_id	showtime_id	booked_id
Edit Copy Delete	0000000001	A05	00004	0000000001
Edit Copy Delete	0000000002	A06	00004	0000000001
Edit Copy Delete	0000000003	B08	00004	0000000002
Edit Copy Delete	0000000004	B09	00004	0000000002
Edit Copy Delete	0000000005	B10	00004	0000000002
Edit Copy Delete	0000000006	E03	00004	0000000003
Edit Copy Delete	0000000007	D03	00004	0000000004
Edit Copy Delete	0000000008	D04	00004	0000000004
Edit Copy Delete	0000000009	D07	00004	0000000005
Edit Copy Delete	0000000010	D08	00004	0000000005
Edit Copy Delete	0000000011	C01	00004	0000000006
Edit Copy Delete	0000000012	C02	00004	0000000006
Edit Copy Delete	0000000013	C10	00004	0000000007
Edit Copy Delete	0000000014	E08	00004	0000000008
Edit Copy Delete	0000000015	E09	00004	0000000008
Edit Copy Delete	0000000016	E10	00004	0000000008
Edit Copy Delete	0000000017	C05	00004	0000000009
Edit Copy Delete	0000000018	C06	00004	0000000009
Edit Copy Delete	0000000019	C07	00004	0000000009
Edit Copy Delete	0000000020	B05	00004	0000000010
Edit Copy Delete	0000000021	A01	00011	0000000011
Edit Copy Delete	0000000022	A02	00011	0000000011

Figure 31 Screenshot of the ticket table in the database after booking two new tickets (A01 and A02)

```php
//insert into ticket
// Need to grab max ID
$query = mysqli_query($conn, "SELECT max(booked_id) AS booked_id FROM booked");
$result = mysqli_fetch_array($query);
$booked_id = $result['booked_id'];
foreach($_POST['checkbox'] as $key => $val){
    $query = mysqli_query($conn, "INSERT INTO ticket (seat_id, showtime_id, booked_id) VALUE ('$val', $showtime_id, $booked_id)");
    if(!$query){
        echo "<br>";
        echo "ERROR: Could not execute query" .mysqli_error($conn);
        echo "<br>";
    }
}

// Update seating availibility
foreach($_POST['checkbox'] as $key => $val){
    $query = mysqli_query($conn, "UPDATE seat SET seat_status = 'B' WHERE seat_id = '$val' AND theatre_id = $theatre_id");
    if(!$query){
        echo "<br>";
        echo "ERROR: Could not execute query" .mysqli_error($conn);
        echo "<br>";
    }
}

if($query){
    echo "<script>
            alert('Ticket(s) created succesfully');
            location='../customer_page/customer_dashboard.php?index=$c_email';
            </script>";
}else{
    echo "<br>ERROR: something overall went wrong. Message admin. <br>";
}
```

Figure 32 Screenshot of the SQL query to insert new booked order into the booked table, and update the status of the seats from available ('A') to booked ('B')

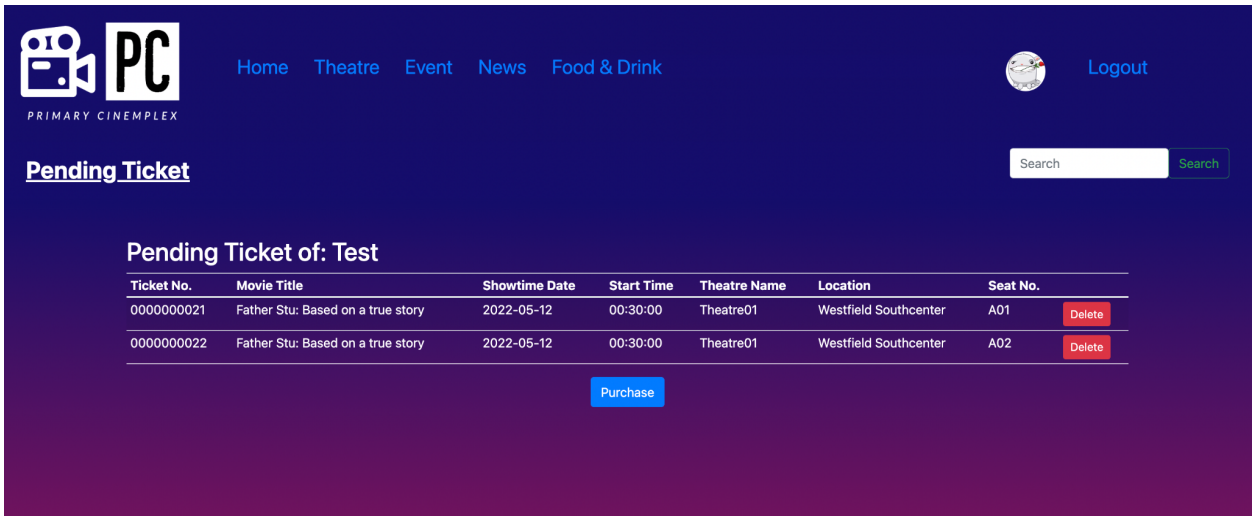

Figure 33 Screenshot of the user dashboard page to show the pending ticket that the customer 'Test' booked

```php
<?php

        $query = mysqli_query($conn, "SELECT * FROM ticket join booked using (booked_id)
        join showtime using (showtime_id)
        join movie using (movie_id)
        join theatre using (theatre_id)
        join branch using (branch_id)
        join seat using (seat_id)
        join customer using (c_email)
        Where c_email = '".$_GET['index']."' and
        booked_status = 'p'
        group by ticket_id");


        while($result = mysqli_fetch_array($query)) {
                $_SESSSION[$result['ticket_id']] = $result['ticket_id'];

?>


        <tr>
                <td><?=$result['ticket_id']?></td>
                <td><?=$result['movie_title']?></td>
                <td><?=$result['showtime_date']?></td>
                <td><?=$result['start_time']?></td>
                <td><?=$result['theatre_name']?></td>
                <td><?=$result['branch_name']?></td>
                <td><?=$result['seat_id']?></td>
                <td><button type="button" class="btn btn-danger btn-sm" style="height: 50%;">Delete</button></td>




        </tr>
<?php }
?>
```

Figure 33 Screenshot of the SQL query to show the pending tickets that the customer 'Test' booked