

Theater System Report

**Introduction**

In the modern society, more and more recreational activities have shown up in people’ life. Film, as one of the most popular entertainment, is largely affecting the transformation of social life. As a result, the benefits of convenient consuming process is playing an essential role in this process. We found the unique advantages of internet by observing the drawbacks of traditional cinema business model, and put forward this Theater System aiming to improve the transaction process and save time and money for people to enjoy movies.

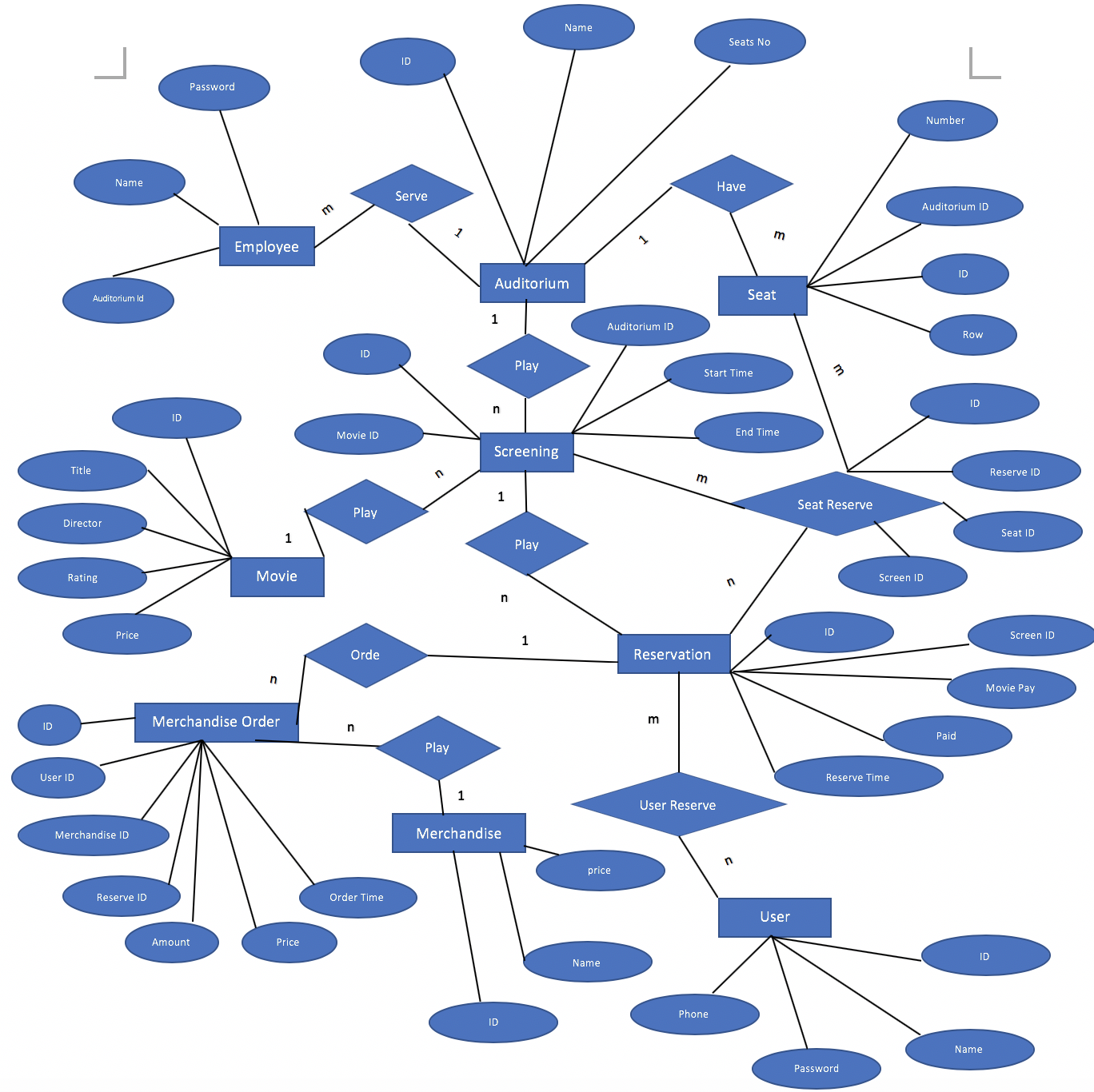
**Types of Users**

This system is about Theatre online booking. In this system, there are two types of users, one is customer who can browse websites, read the list of films information, reserve seats and purchase merchandise, another one is administrators or employees who can add, delete, change, search the data in system, and Accordingly, the customers and administrators have different permissions to access this system based on their account number and password.

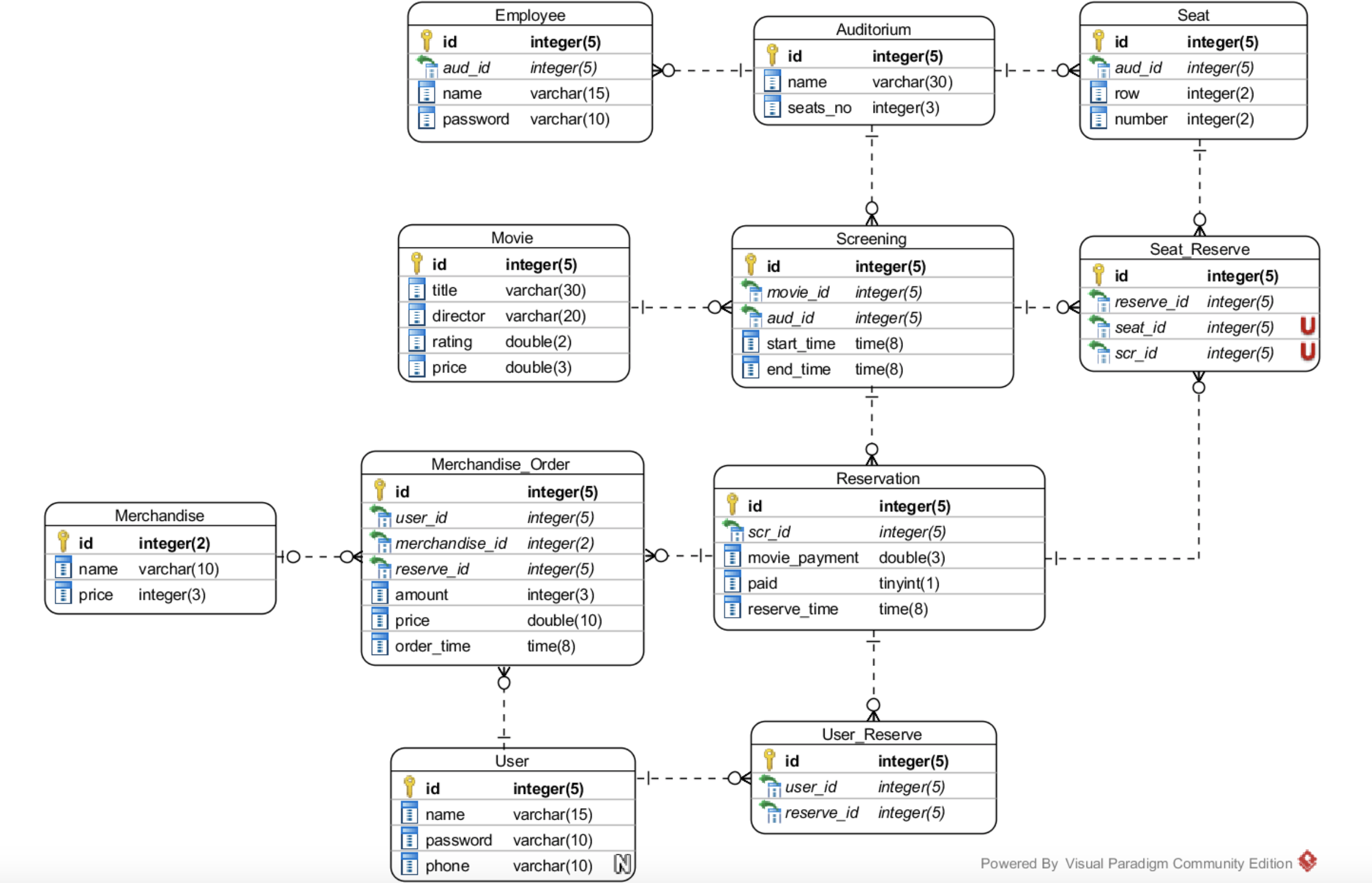
**Assumptions**

Our movie theater system will include:  
• The reservation/withdraw of seats  
• The list of films (time, auditorium, rate, price, etc) for customers  
• The list of employees to each auditorium  
• The list of merchandise sold to customers  
• The view of profits per month and the rank of film market

**E-R Diagram**



**Relational schemas**



**DDL statement**

1. **Table structure for table `auditorium`**

CREATE TABLE IF NOT EXISTS `auditorium` (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`name` varchar(30) NOT NULL,

`seats\_no` int(3) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=6 ;

1. **Table structure for table `employee`**

CREATE TABLE IF NOT EXISTS `employee` (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`name` varchar(15) NOT NULL,

`password` varchar(10) NOT NULL,

`aud\_id` int(5) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `aud\_id` (`aud\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

1. **Table structure for table `merchandise`**

CREATE TABLE IF NOT EXISTS `merchandise` (

`id` int(2) NOT NULL AUTO\_INCREMENT,

`name` varchar(10) NOT NULL,

`price` int(3) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=5 ;

1. **Table structure for table `merchandise\_order`**

CREATE TABLE IF NOT EXISTS `merchandise\_order` (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`user\_id` int(5) NOT NULL,

`merchandise\_id` int(2) DEFAULT NULL,

`reserve\_id` int(5) NOT NULL,

`amount` int(3) NOT NULL DEFAULT '0',

`price` double NOT NULL,

`order\_time` datetime NOT NULL,

PRIMARY KEY (`id`),

KEY `merchandise\_id` (`merchandise\_id`),

KEY `user\_id` (`user\_id`),

KEY `reserve\_id` (`reserve\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=1 ;

1. **Table structure for table `movie`**

CREATE TABLE IF NOT EXISTS `movie` (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`title` varchar(50) NOT NULL,

`director` varchar(20) NOT NULL,

`rating` double NOT NULL,

`price` double NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=109 ;

1. **Table structure for table `reservation`**

CREATE TABLE IF NOT EXISTS `reservation` (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`scr\_id` int(5) NOT NULL,

`movie\_payment` double NOT NULL,

`paid` tinyint(1) NOT NULL,

`reserve\_time` datetime NOT NULL,

PRIMARY KEY (`id`),

KEY `scr\_id` (`scr\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=20 ;

1. **Table structure for table `screening`**

CREATE TABLE IF NOT EXISTS `screening` (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`movie\_id` int(5) DEFAULT NULL,

`aud\_id` int(5) DEFAULT NULL,

`start\_time` datetime DEFAULT NULL,

`end\_time` datetime DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `movie\_id` (`movie\_id`),

KEY `aud\_id` (`aud\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=71 ;

1. **Table structure for table `seat`**

CREATE TABLE IF NOT EXISTS `seat` (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`row` int(2) NOT NULL,

`number` int(2) NOT NULL,

`aud\_id` int(5) NOT NULL,

PRIMARY KEY (`id`),

KEY `aud\_id` (`aud\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=211 ;

1. **Table structure for table `seat\_reserve`**

CREATE TABLE IF NOT EXISTS `seat\_reserve` (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`reserve\_id` int(5) NOT NULL,

`seat\_id` int(5) NOT NULL,

`scr\_id` int(5) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `scr\_id\_2` (`scr\_id`,`seat\_id`),

KEY `reserve\_id` (`reserve\_id`),

KEY `seat\_id` (`seat\_id`),

KEY `scr\_id` (`scr\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=14 ;

1. **Table structure for table `user`**

CREATE TABLE IF NOT EXISTS `user` (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`name` varchar(15) NOT NULL,

`password` varchar(10) NOT NULL,

`phone` varchar(10) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=4 ;

1. **Table structure for table `user\_reserve`**

CREATE TABLE IF NOT EXISTS `user\_reserve` (

`id` int(5) NOT NULL AUTO\_INCREMENT,

`user\_id` int(5) DEFAULT NULL,

`reserve\_id` int(5) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `user\_id` (`user\_id`),

KEY `reserve\_id` (`reserve\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=18 ;

1. **Constraints for table `employee`**

ALTER TABLE `employee`

ADD CONSTRAINT `employee\_ibfk\_2` FOREIGN KEY (`aud\_id`) REFERENCES `auditorium` (`id`) ON DELETE SET NULL ON UPDATE CASCADE;

1. **Constraints for table `merchandise\_order`**

ALTER TABLE `merchandise\_order`

ADD CONSTRAINT `merchandise\_order\_ibfk\_1` FOREIGN KEY (`merchandise\_id`) REFERENCES `merchandise` (`id`) ON DELETE SET NULL ON UPDATE CASCADE,

ADD CONSTRAINT `merchandise\_order\_ibfk\_2` FOREIGN KEY (`user\_id`) REFERENCES ` user` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `merchandise\_order\_ibfk\_3` FOREIGN KEY (`reserve\_id`) REFERENCES `reservation` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

1. **Constraints for table `reservation`**

ALTER TABLE `reservation`

ADD CONSTRAINT `reservation\_ibfk\_1` FOREIGN KEY (`scr\_id`) REFERENCES `screening` (`id`) ON DELETE CASCADE;

1. **Constraints for table `screening`**

ALTER TABLE `screening`

ADD CONSTRAINT `screening\_ibfk\_3` FOREIGN KEY (`movie\_id`) REFERENCES `movie` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `screening\_ibfk\_4` FOREIGN KEY (`aud\_id`) REFERENCES `auditorium` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

1. **Constraints for table `seat`**

ALTER TABLE `seat`

ADD CONSTRAINT `seat\_ibfk\_1` FOREIGN KEY (`aud\_id`) REFERENCES `auditorium` (`id`) ON DELETE CASCADE;

1. **Constraints for table `seat\_reserve`**

ALTER TABLE `seat\_reserve`

ADD CONSTRAINT `seat\_reserve\_ibfk\_1` FOREIGN KEY (`reserve\_id`) REFERENCES `reservation` (`id`) ON DELETE CASCADE,

ADD CONSTRAINT `seat\_reserve\_ibfk\_2` FOREIGN KEY (`seat\_id`) REFERENCES `seat` (`id`) ON DELETE CASCADE,

ADD CONSTRAINT `seat\_reserve\_ibfk\_3` FOREIGN KEY (`scr\_id`) REFERENCES `screening` (`id`) ON DELETE CASCADE;

1. **Constraints for table `user\_reserve`**

ALTER TABLE `user\_reserve`

ADD CONSTRAINT `user\_reserve\_ibfk\_5` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`) ON DELETE SET NULL ON UPDATE CASCADE,

ADD CONSTRAINT `user\_reserve\_ibfk\_6` FOREIGN KEY (`reserve\_id`) REFERENCES `reservation` (`id`) ON DELETE SET NULL ON UPDATE CASCADE;

**Front-End**

About front-end design, we used Bootstrap and JavaScript to design and implement our system, by importing the fixed format, navigation, table and so on, it saves lots of time for us to develop our system.

The specific process of connection between front-end and back-end

**Connection of Front-End and Back-End**

In this system, we used SQL statement and PHP to build connection between Front-End and Back-End.

**System Design and Implementation**

In this part, we will introduce our system implementation process by describing various functions of The Theater System.

1. Register and log in
2. Browsing information
3. Reserve a seat
4. Purchase merchandises
5. Manage account

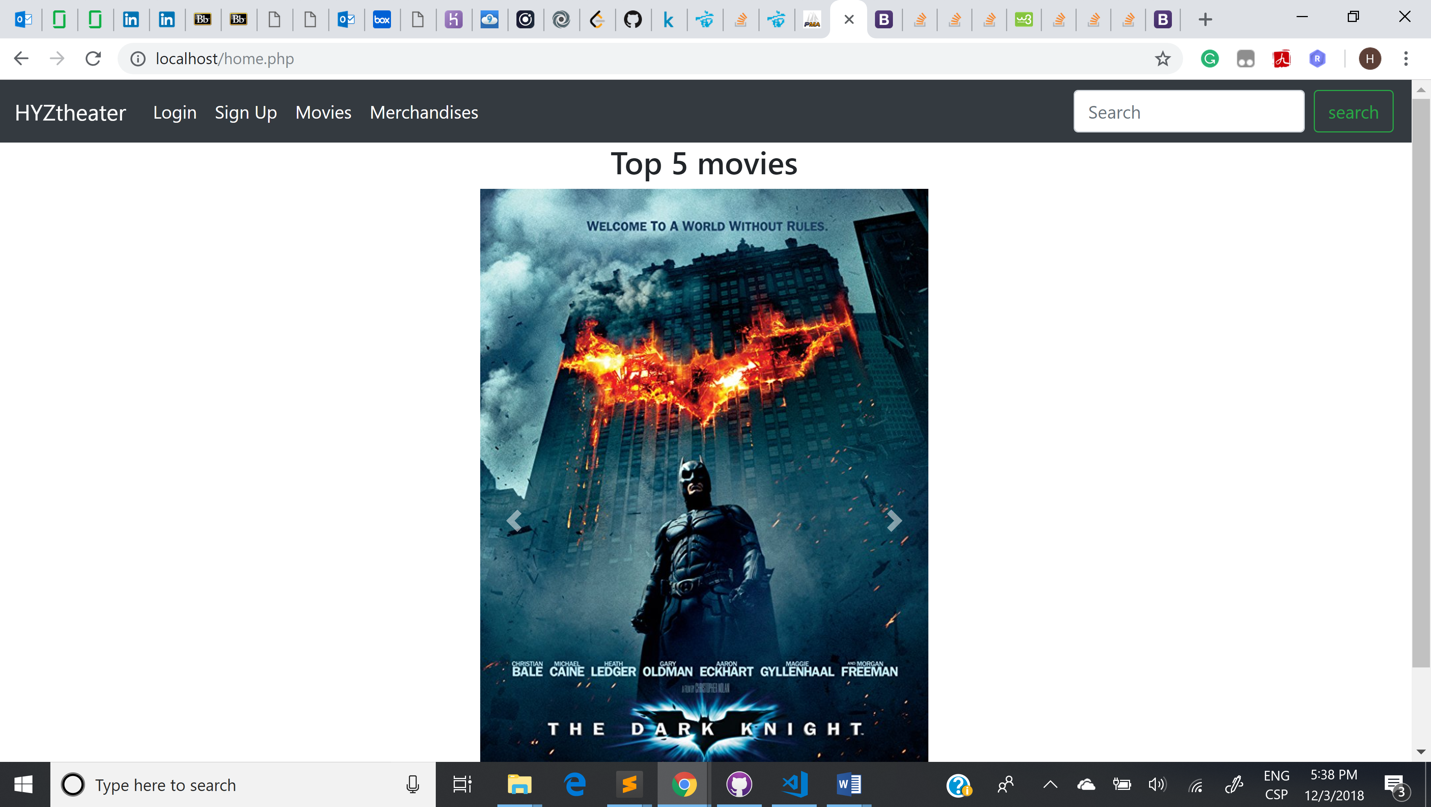


Fig1. Home Page

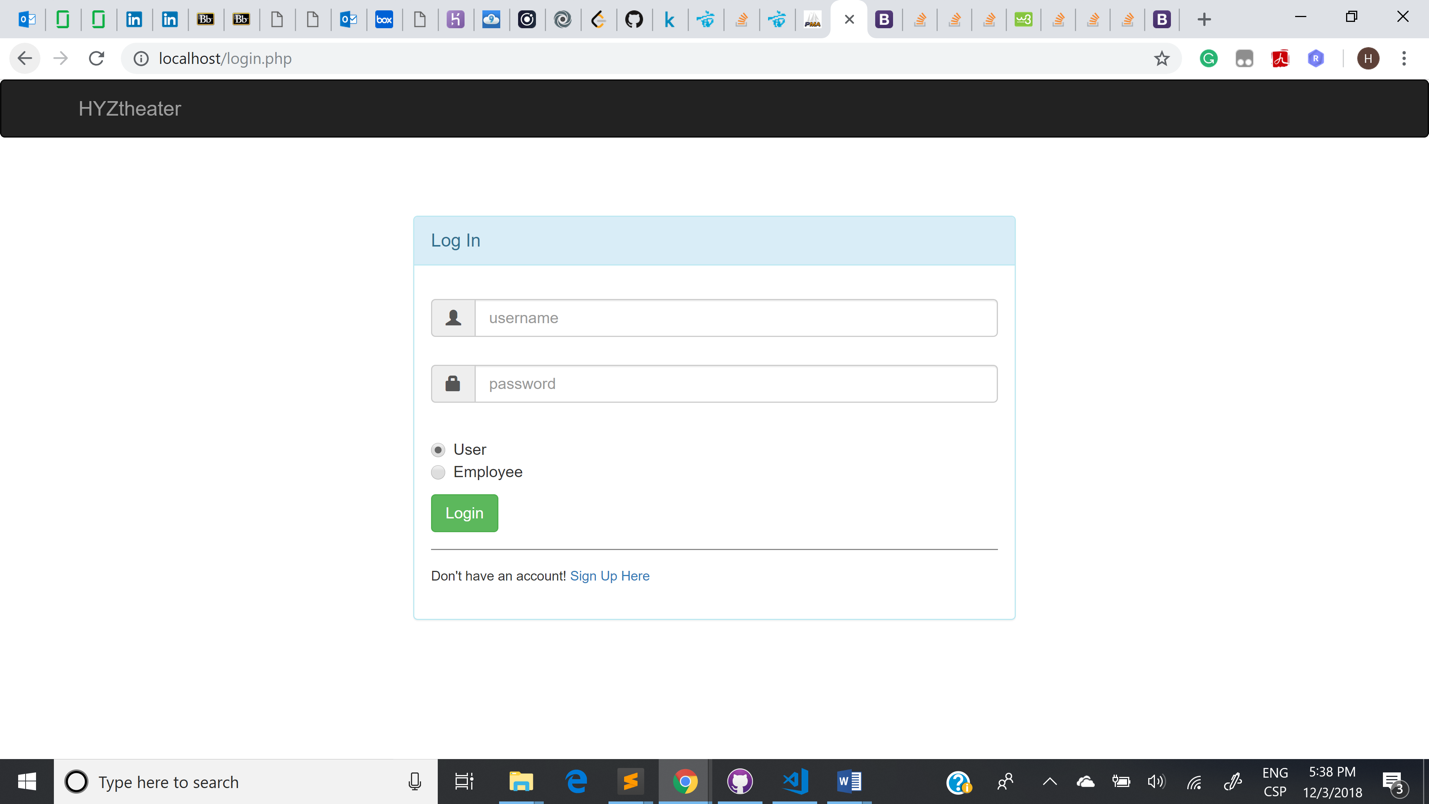


Fig2. Log In

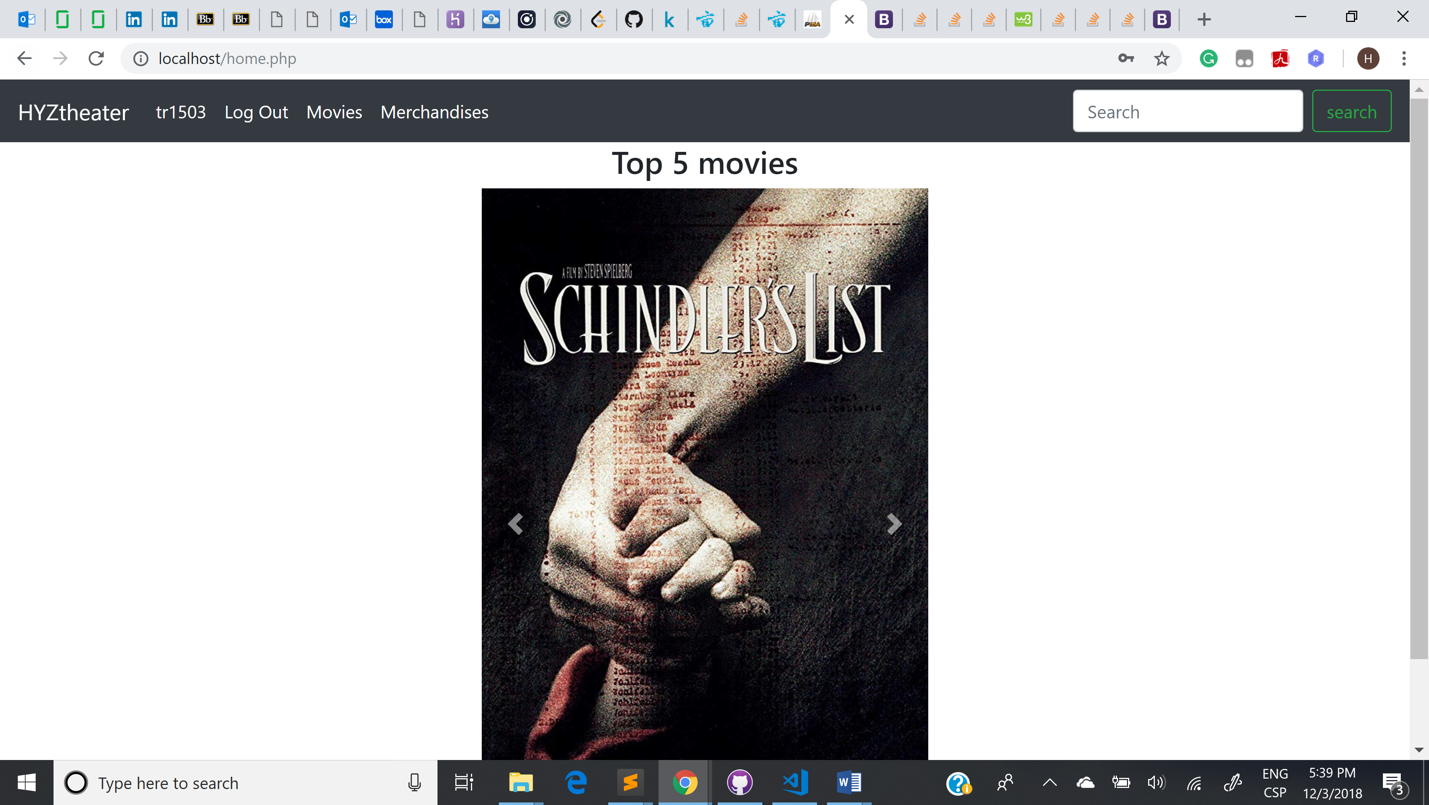


Fig3. Log in successfully

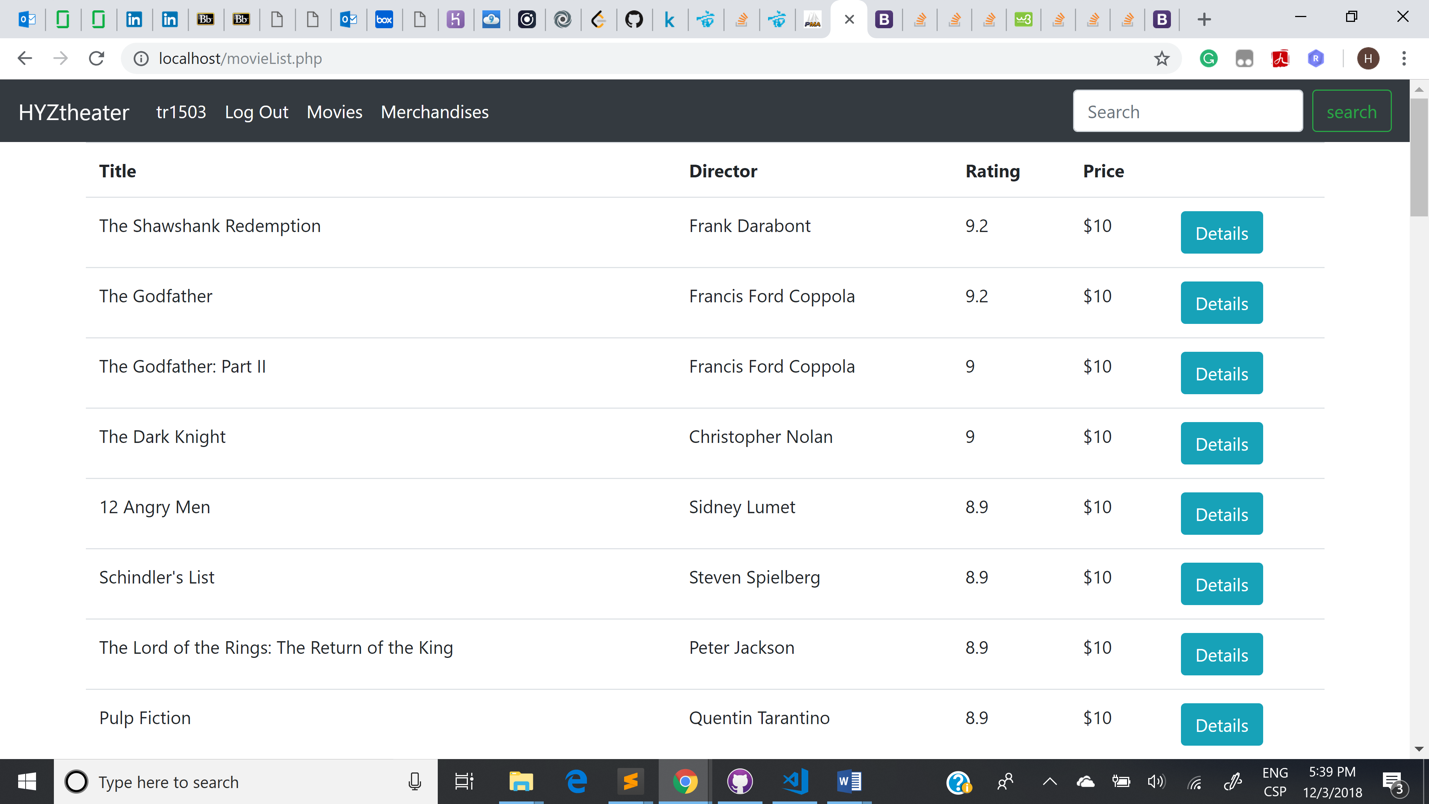


Fig4. Movie Lists

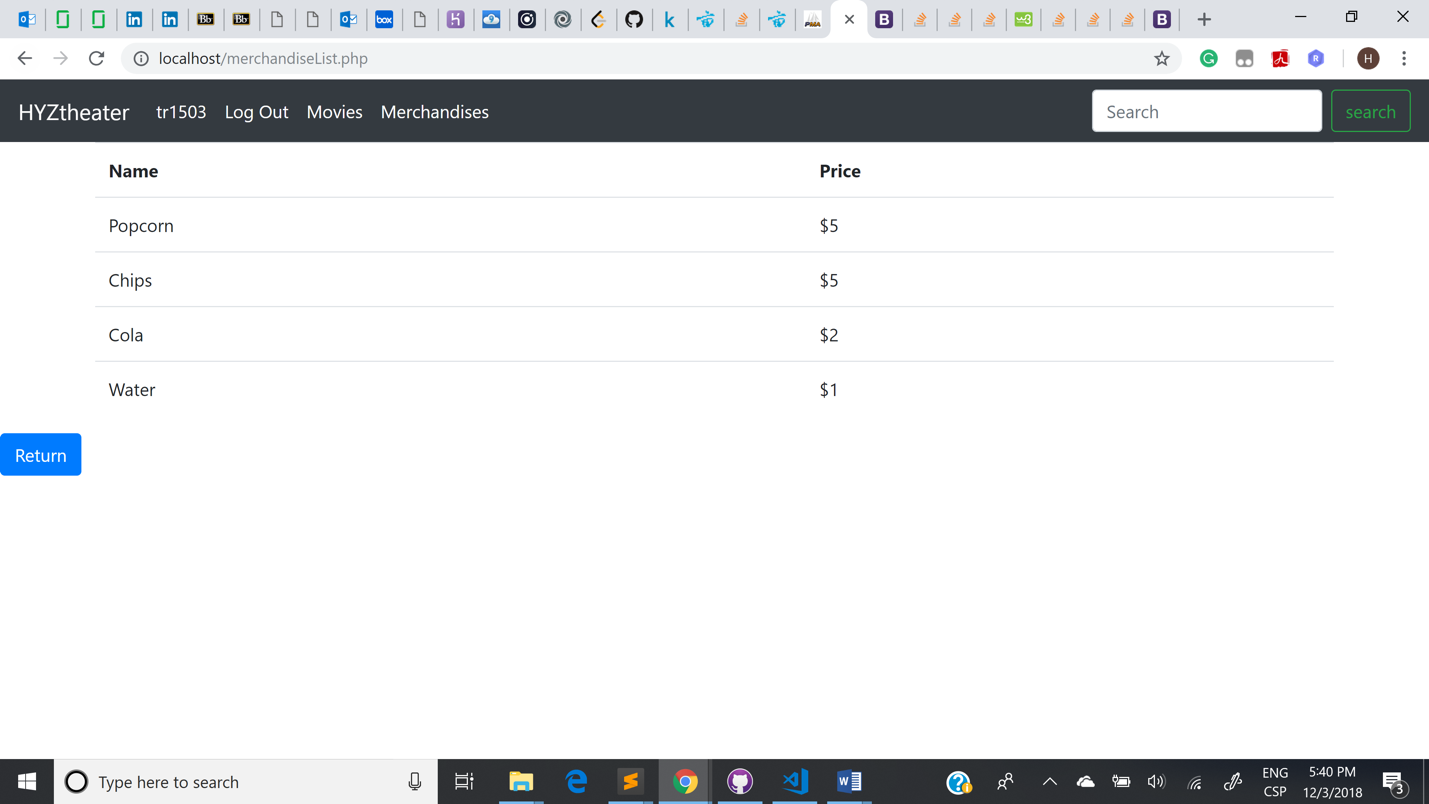


Fig5. Merchandises Lists

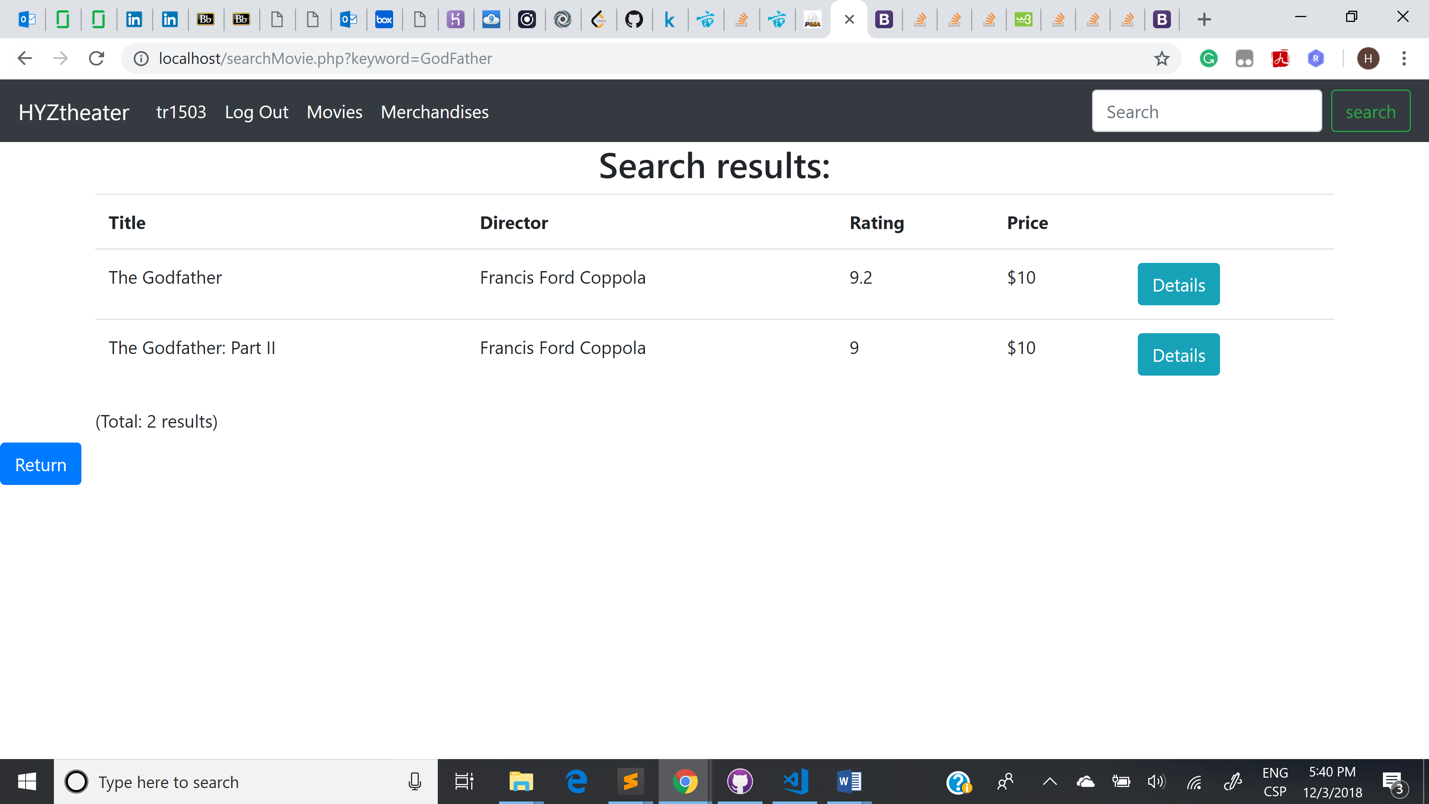


Fig6. Search Movies

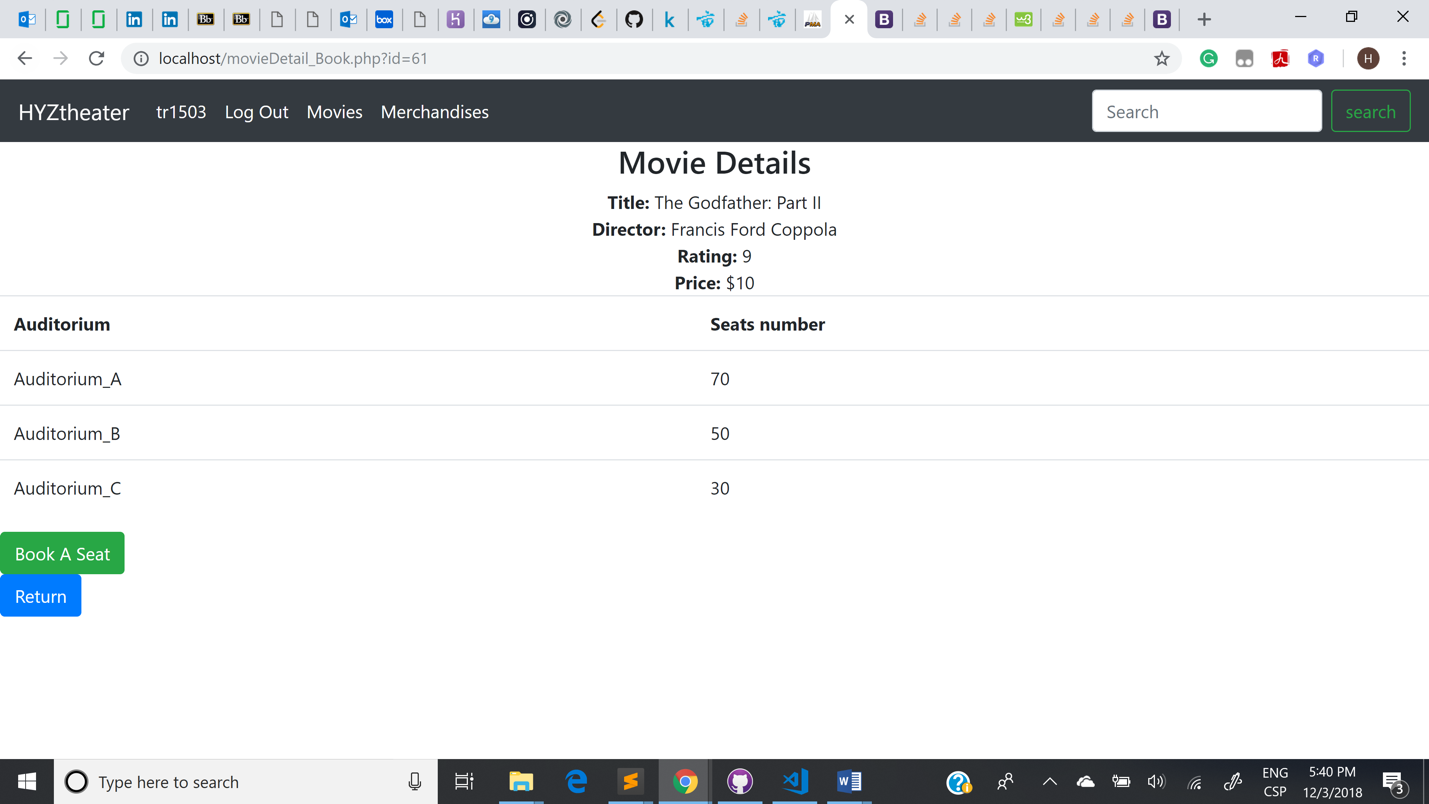


Fig7. Screening Details

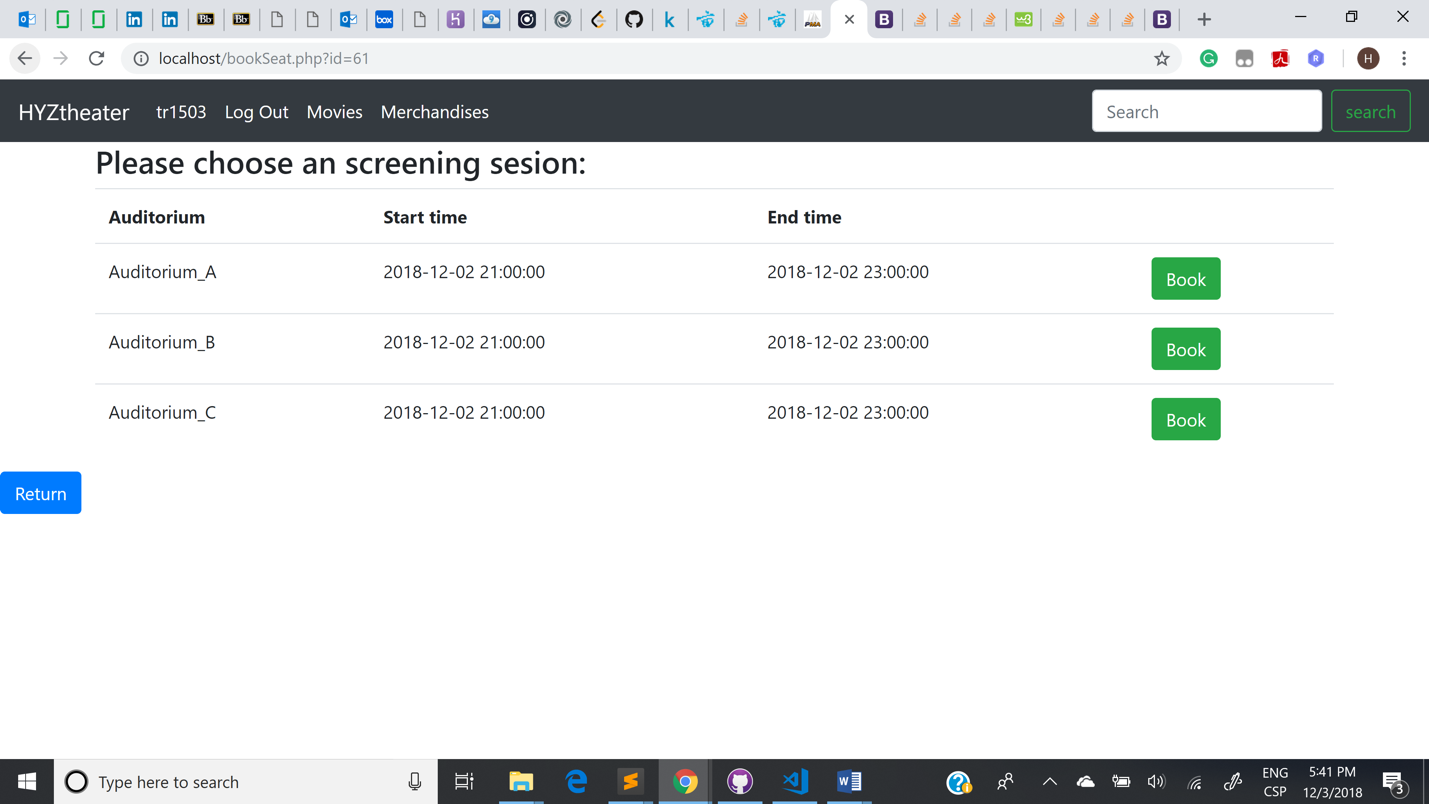


Fig8. Select Screening Session

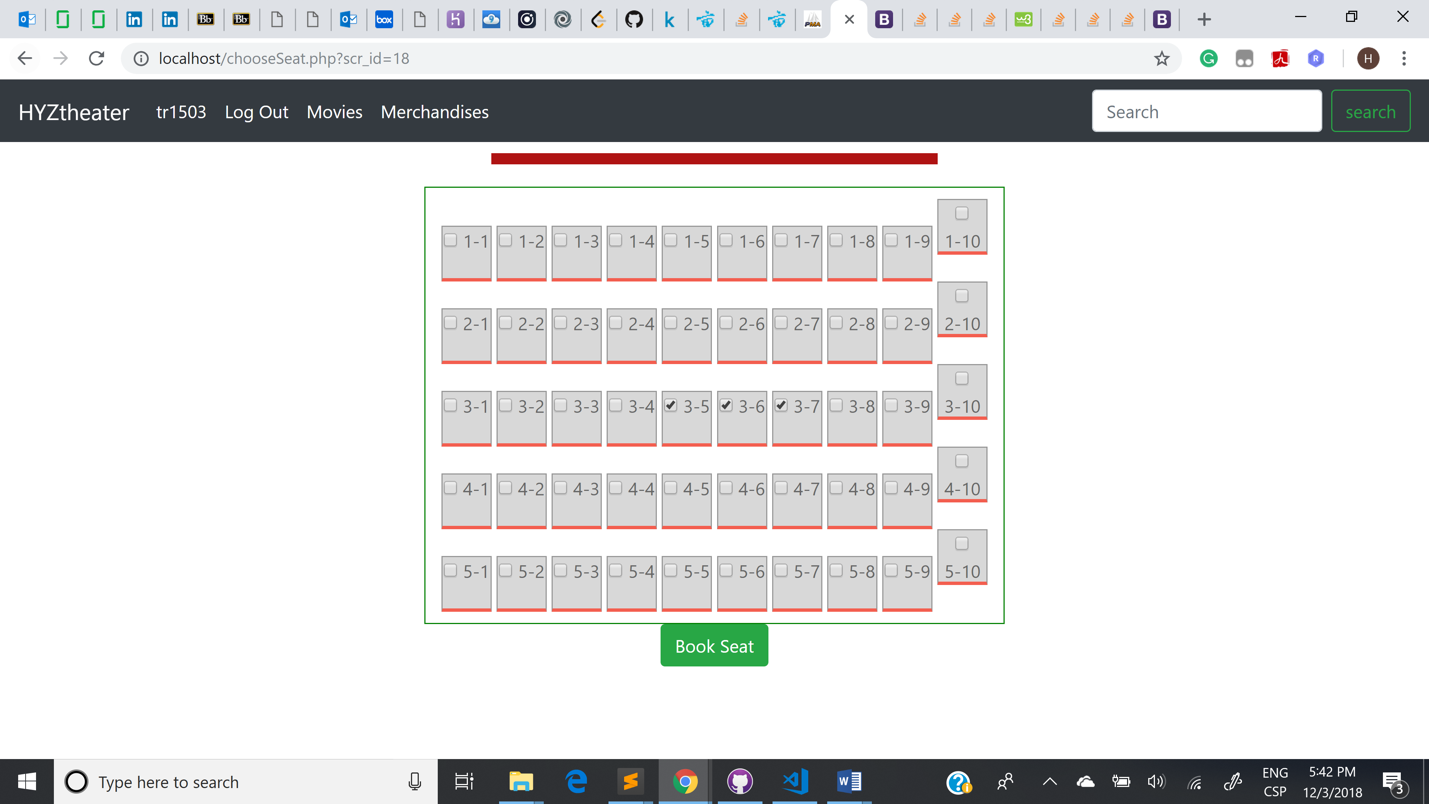


Fig9. Auditorium Layout

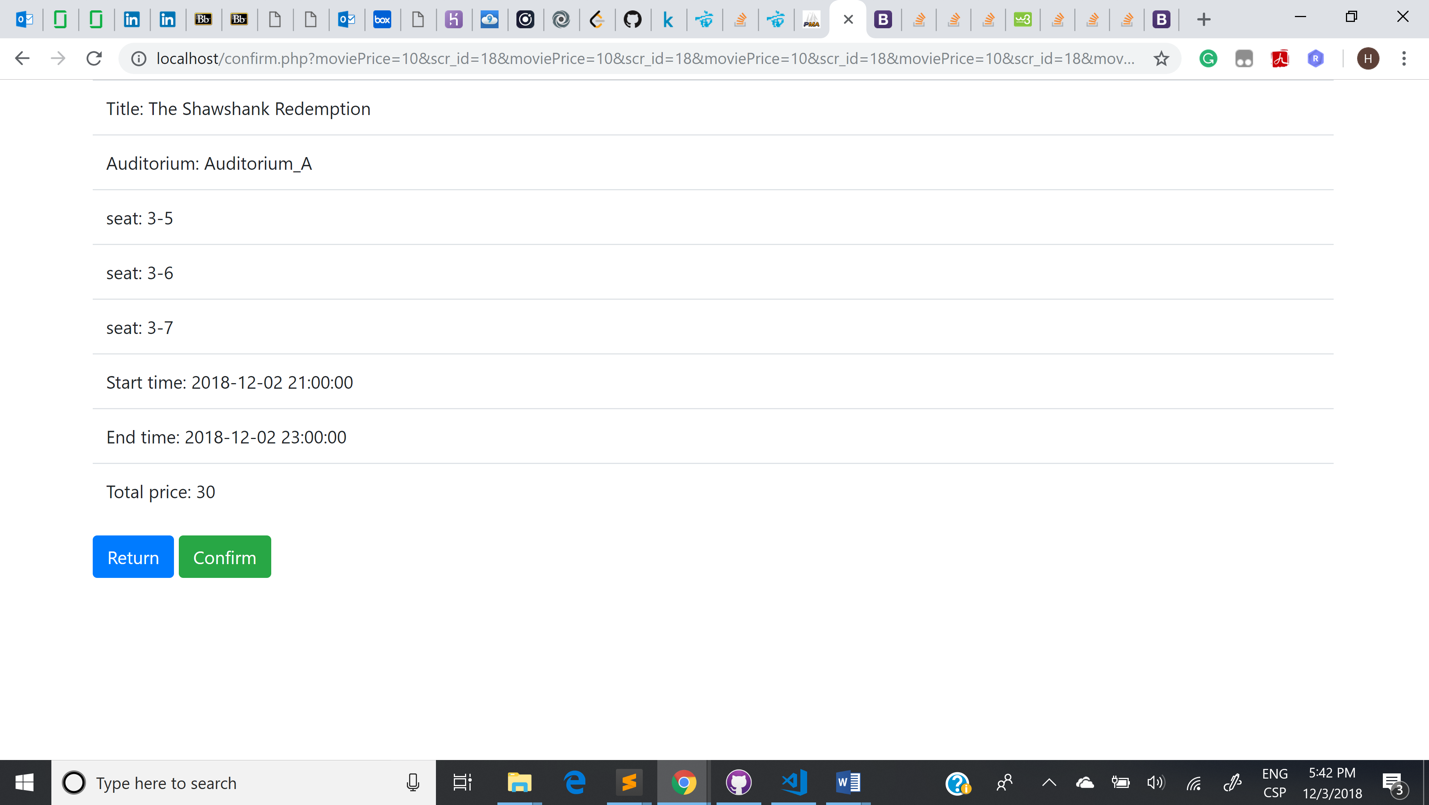


Fig10. Order Details

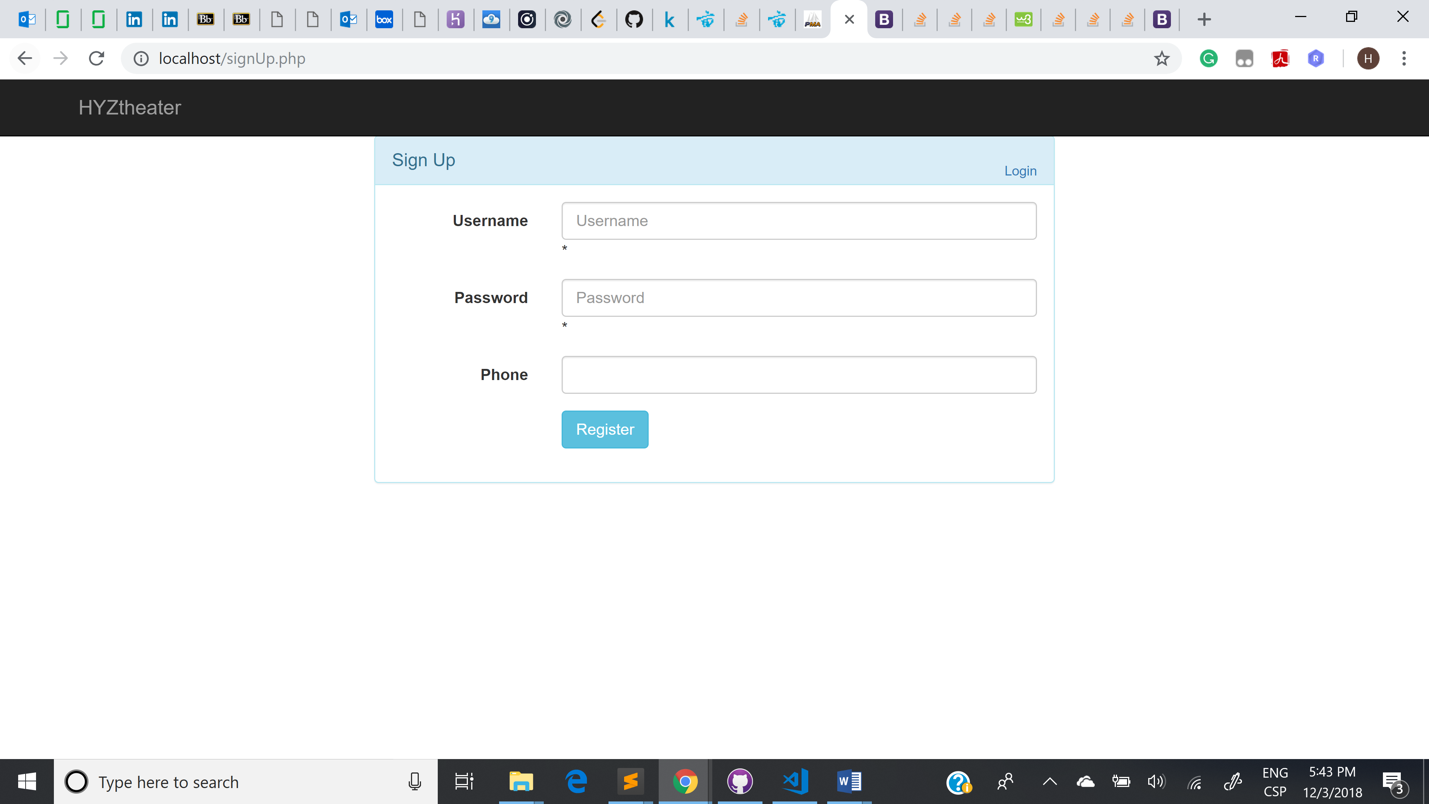


Fig11. Register



Fig12. Employee Home Page

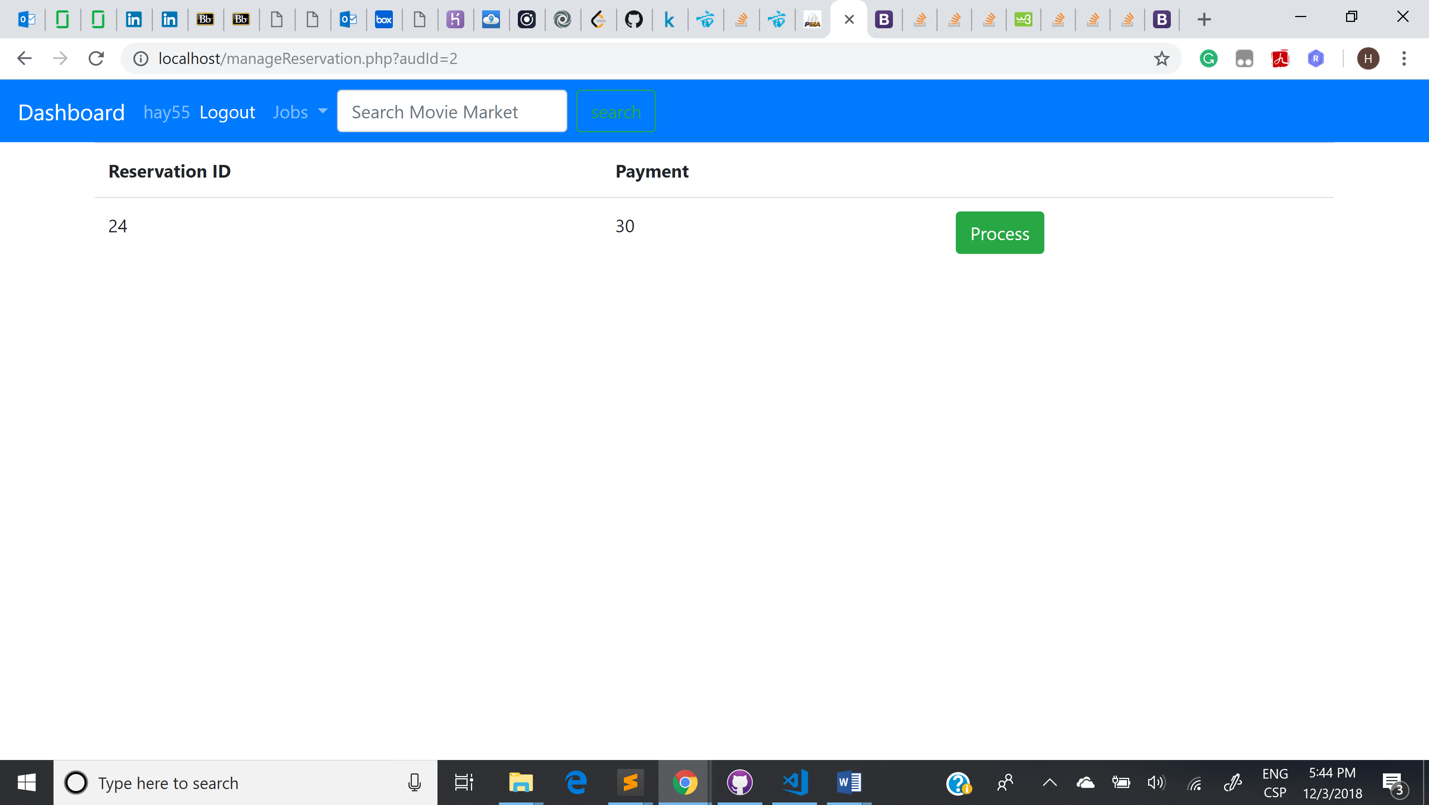


Fig13. Managing Reservation



Fig14. After process of reservation

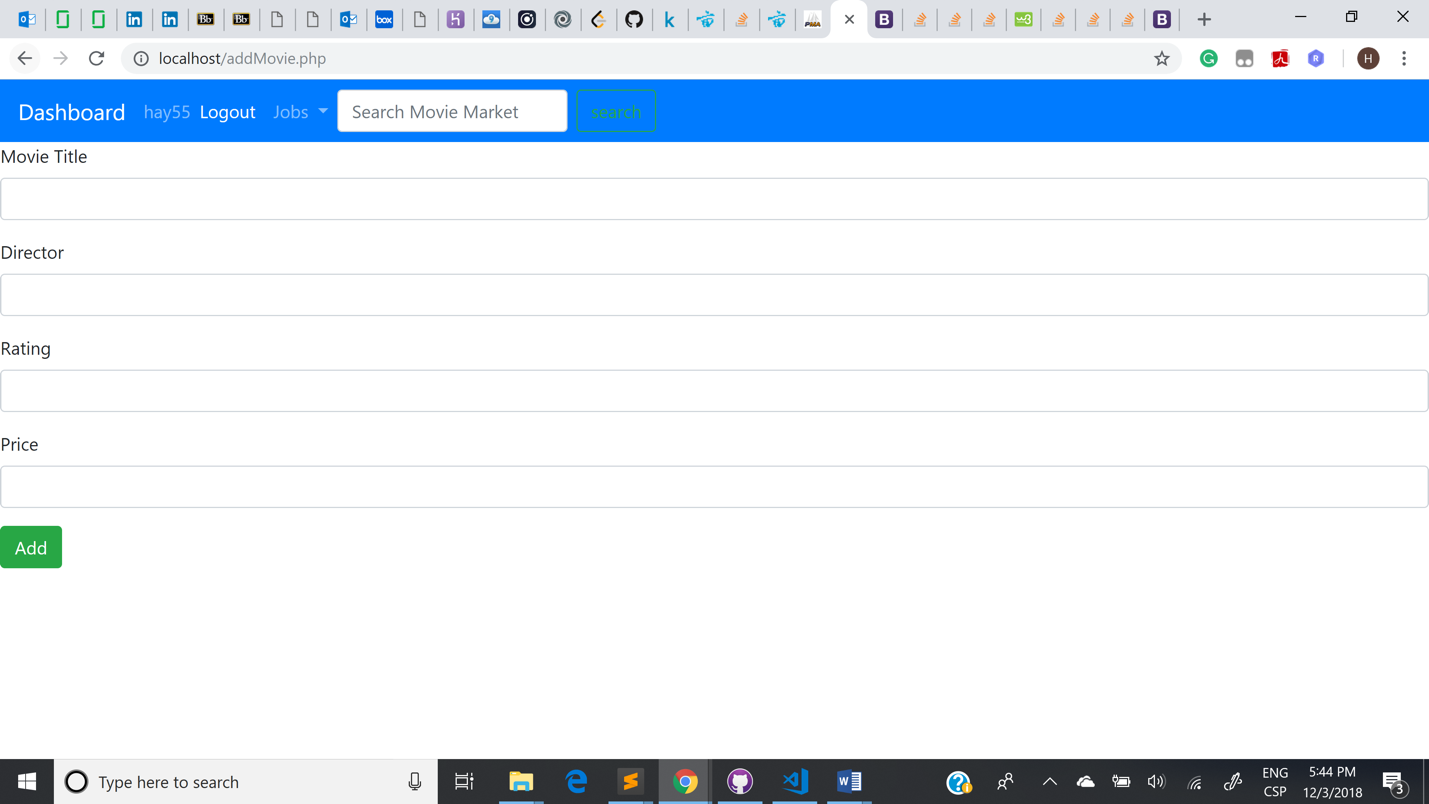


Fig15. Add movie

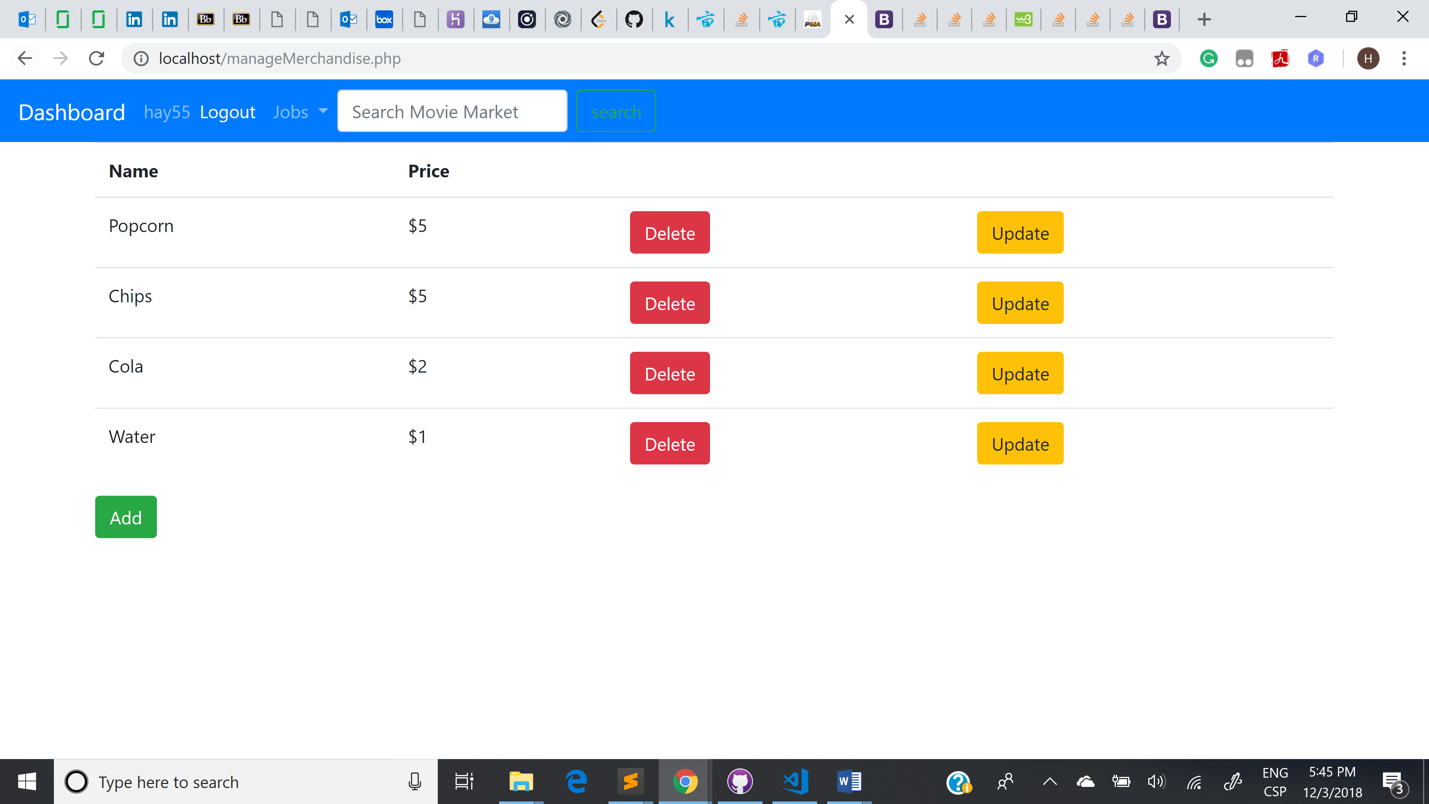
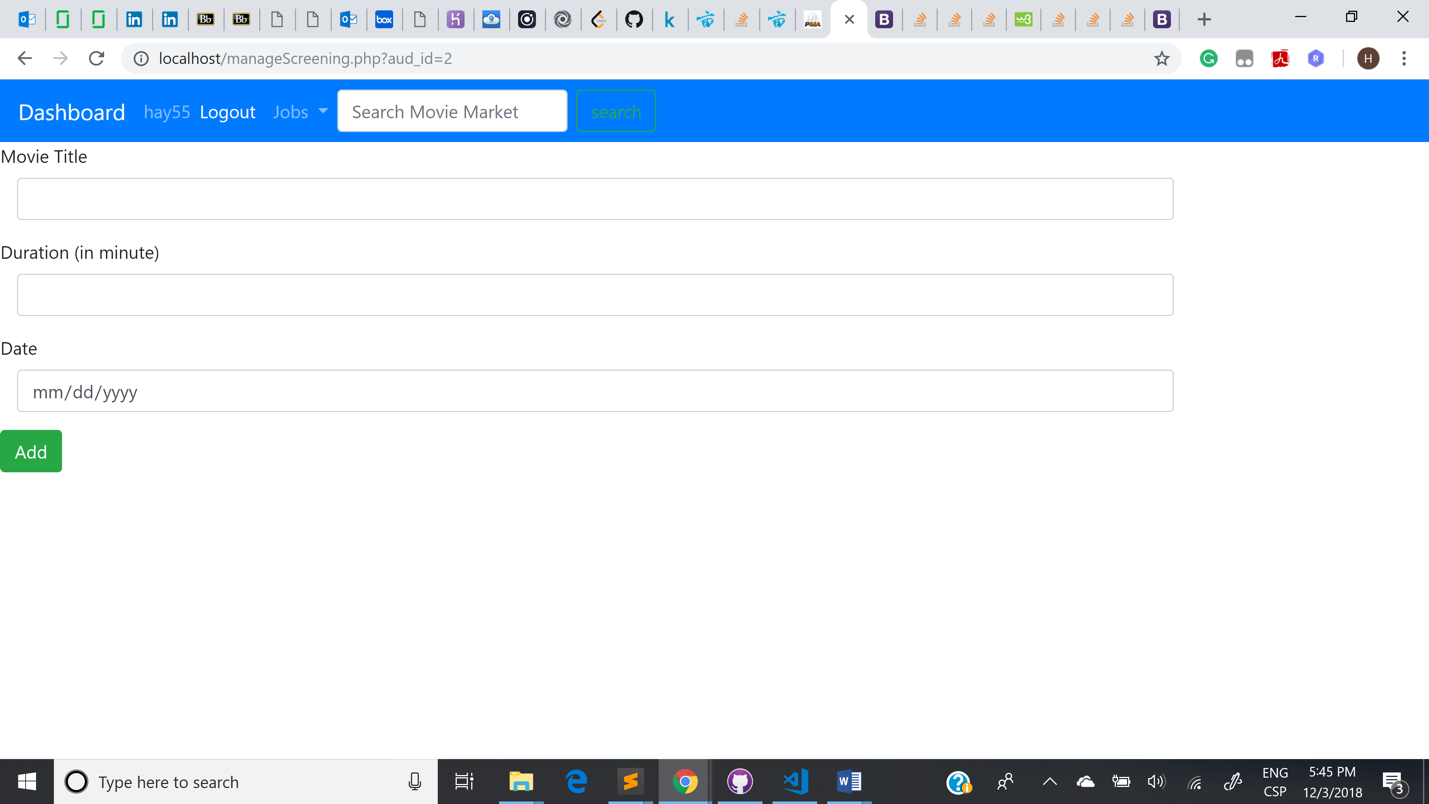


Fig16. Manage Merchandises



Fig17. Choose Screening

  
Fig18. Add Screening

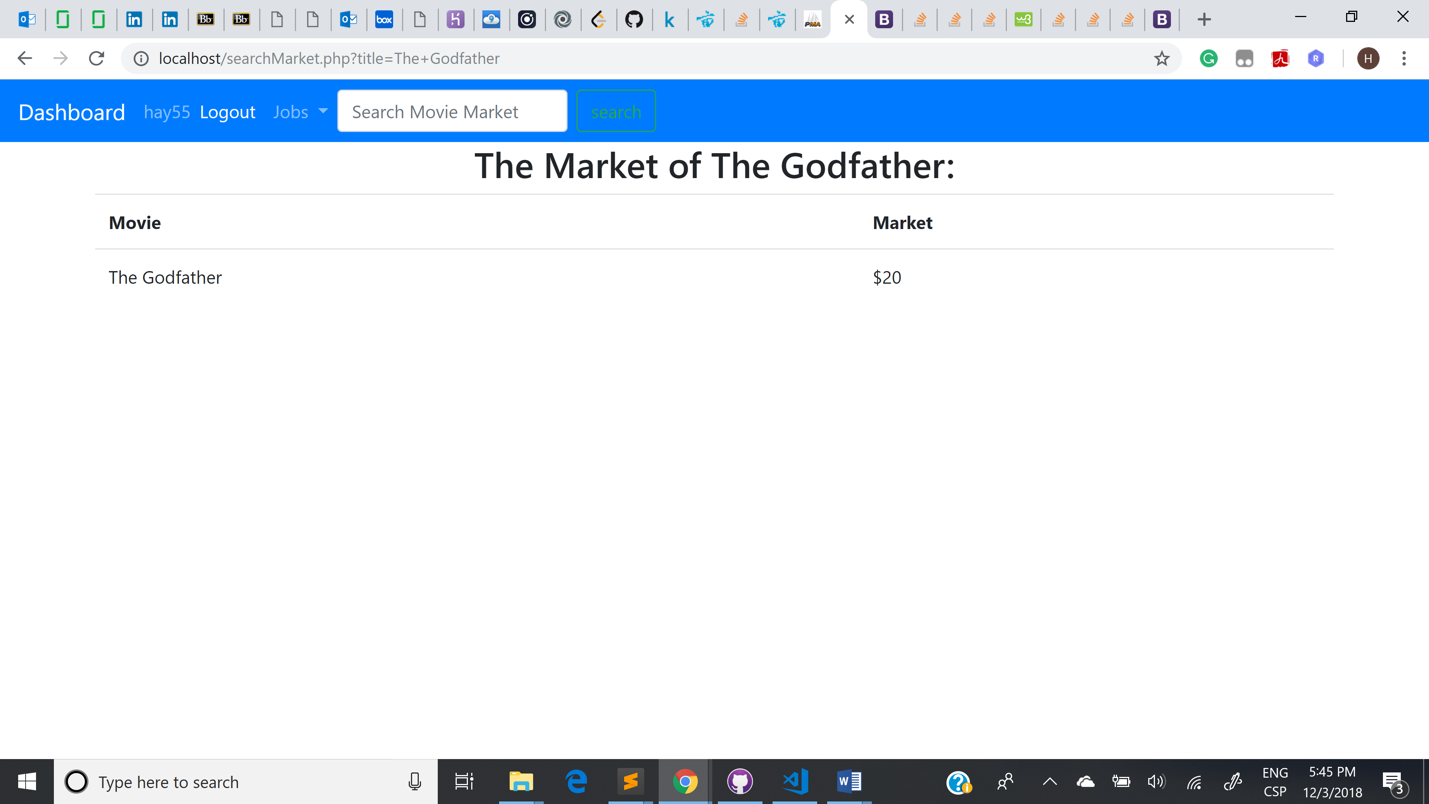


Fig19. Search Marketing

**Testing and erroneous cases**

An error may occur when customer reserve a seat in Theater system, as it has been occupied by others.

Testing process used in this system is mainly about logging in and registering test, data operations test.

In the test process of registering, we used different characters or number as input to check information returned.

In the process of logging in, we used two types of users, such as customers and employees, to check the information of success or failure.

Data operations test is about four kinds of operations, adding, deleting, changing, search.

According to system model, we test these data operations by checking different functions designed in this system.

First, adding test. In this section, we test data of movies and merchandise. The reservation will increase as customers select and reserve a seat in theater system.

Second, deleting test, In this section, employee can delete the data in this system.

Third, changing test, in this section, employee can change information in system.

Lastly, search test, by using search function, customers can get information of related movies and merchandise.

**Limitations and Improvements**

1. **Limitations:**

In this system, front-end is not great and convenient enough to use or manage. For example, customers can not select their seats in theater easily by a two-dimensional or three-dimensional diagram.

**Improvement:**

Design and implement a two-dimensional or three-dimensional diagram.

1. **Limitations:**

The function of selecting merchandise have not been implemented fully.

**Improvements:**

Build more tables and use PHP to finish this process of adding, deleting, updating, searching.

1. **Limitations:**

About data, we did not obtain enough data information and manipulate data in large scale.

**Improvement:**

In the next step, import more data information, and use them to test the capacity and speed of this system when people use it.

1. **Limitations:**

Administer should have permissions to manage the employees working in Theater. And employees should be responsible for managing the merchandise. So, there should be three types of users, they are customers, administers, employees.

**Improvement:**

Create a new entity named administrator who have the highest authority to manage the whole system, including add, delete, change, search the information of employees while employees can manage the data information of reservation and movies and merchandise and so on.

1. **Limitations:**

There is no real payment process, in this system, all processes of transaction are only involved in operations of database.

**Improvement:**

Design and implement the payment process, which can be used in user’s account.

1. **Limitations:**

When customers reserve seats, the same seat can not be selected repeatedly, but there is no signal and reminder for users.

**Improvements:**

This problem can be resolved by using JavaScript to add special effects for users to know if the seats have been occupied.