****

**A Web-based Video Rental Site**

**Rent A Movie**

**Project Team:**

Brett Davis, Marvin Jenkins, Jeffrey Li

Corey McCoy, Christopher Poole, Michael Powell and Ward Williams

**Contributions**

Michael Powell: PHP Programming (Database Connection and Backend Functionality), MySQL Server Setup, Web Server Setup, and Minor Front End Contributions

Marvin Jenkins: Database Design and Implementation, Queries, Integration Testing

Ward Williams: Front End Designing/Developing (HTML/CSS Programming), Minor Back End Contributions.

Chris Poole: Documentation, Minor Front End Design, Bug testing and correction

Jeffrey Li: Database Design/Implementation, Referential Integrity Rules, Password Security

Brett Davis: Front End Designing/Developing (HTML/CSS), Use Cases, Bug Testing, Data Entry

Corey McCoy: Documentation,Testing and Presentation

CSCI 5436 Distributed Web Systems Design

Department of Computer Science

Georgia Southern University

Spring 2018

**Table Of Contents**

Introduction……...………………………………………………………………………...3

Project Overview…..……………………………………………………………………...3

Work Breakdown..………………………………………………………………………...3

Use Cases…………..……………………………………………………………………...4

Diagrams…………....………………………………………………………………….....9

ER Diagram…………………………...………………………………………….9

Database Diagram…………………...…………………………………………...10

Website Design…………………………………………………………………………..12

HTML Code……………………………………………………………………………...15

Database Code/Queries…………………………………………………………………..19

PHP……………………………………………………………………………………....24

Site Structure……………………………………………………………………………..34

Testing……………………………………………………………………………………35

Conclusion……………………………………………………………………………….38

**Introduction**

Our project is a web-based video rental website named Rent A Movie which allows users to buy, rent and view movies online. It will also allow customers to view a list of which movies are available on the site. This project will consist of : a website to display the movies and allow users to input their information and a database to store this information securely as well as keep a record of movies located on the site.

Site Link:<http://18.216.85.101>

**Project Overview**

The goal of this project was to create a website that could simulate the functions of a movie streaming site such as Netflix. We also wanted our website to be appealing and easy to navigate for the user. We implemented our webpage using HTML,AJAX and PHP.

On the database end we used MySQL with PHP to interact with it. The tables in the database are: User, Currently\_watching, History and Movies. Our User table contains the users email address and phone number and id of movies owned. The History table has the id’s of the movies previously rented. The movies table has the name an id of all movies on the site. Lastly, Currently\_Watching contains the id of movies currently being watched.

**Work Breakdown**

Our project team has 7 members. To complete this project effectively we decided to split into 3 groups. Our groups looked like this:

**Database Design:** Marvin Jenkins, Jeffrey Lee, Mike Powell

**Web Server Design & HTML:** Brett Davis, Ward Williams, Mike Powell

**Presentation, Documentation, & CSS:** Corey McCoy, Chris Poole, Mike Powell

The group leader was Mike Powell who divided the team into these roles based on whose skills fit most with the job. We worked in stages to complete this task. the first stage was planning the design of the database and site as well as making use cases. Next was creation of the site and table followed by testing and lastly documentation.

**Use Cases**

**Case number:** RM 1.1

**Title (Goal):** Open webpage

**Description:** User connects to web page with a browser.

**Primary Actors:** User, Browser, Page server

**Main:** Pre-condition: User has a computer on with valid browser installed.

1. User enters address or clicks on shortcut directed to webpage.

2. Browser sends request to page server for start page.

3. Page server receives request and sends template page to browser.

E3.1 Page server is unable to locate a page.

E3.1.1 Page server sends ‘404 error’ to browser for template or individual page within template.

4. Template page sends request through browser for login, header, base and about pages.

E4.1 Page server is unable to locate a page.

E4.1.1 Page server sends ‘404 error’ to browser for template or individual page within template.

5. Browser displays pages in appropriate sections of template page.

Post condition: User may continue to use case RM 1.2 or RM 1.3.

**Case number:** RM 1.2

**Title (Goal):** Create account

**Description:** User creates an account.

**Primary Actors:** User, Browser, Page server, Data server

**Main:** Pre-condition: User has connected to webpage via browser.

1. User clicks create account in login area.

2. Browser sends request to the page server for ‘register.html’.

3. Page server receives request and sends ‘register.html’.

E3.1 Page server is unable to locate ‘register.html’.

E3.1.1 Page server sends ‘404 error’ to browser.

4. Browser receives and displays ‘register.html’ in main section of template page.

5. User enters: First Name, Last Name, Email, Password, Repeat Password, Credit Card Number, Expiration Date, and CVV. Then clicks the ‘submit’ button.

6. Browser posts data to page server.

7. Page server receives and verifies data.

E7.1 User has omitted or not followed correct format on one or more entry items.

E7.1.1 Page server detects error and returns error messages to browser.

E7.1.2 Browser displays ‘register.html’ with active error messages.

E7.1.3 Return to step 5.

8. Page server sends SQL ‘INSERT’ statement for the ‘User’ table to data server.

E8.1 Page server is unable to connect to data server or table.

E8.1.1 Page server receives an error 2002 (no such file or directory) or error 1146 (no such table)

9. Page server sends ‘signIn.html’ to browser

10. Browser displays ‘signIn.html’ in main section of template page.

Post condition: User may continue to use case RM1.3 step 5

**Case number:** RM 1.3

**Title (Goal):** Login account

**Description:** User logs in to their account.

**Primary Actors:** User, Browser, Page server, Data server

**Main:** Pre-condition: User has connected to webpage via browser. Browser is displaying start page.

1. User clicks ‘Sign In’ in login area.

2. Browser sends request to the page server for ‘login.html’.

3. Page server receives request and sends ‘login.html’ to browser.

E3.1 Page server is unable to locate ‘login.html’.

E3.1.1 Page server sends ‘404 error’ to browser.

4. Browser receives and displays ‘login.html’ in main section of template page.

5. User enters their Email, Password and clicks the ‘Sign In’ button.

6. Browser posts data to page server.

7. Page server receives and verifies data.

E7.1 User has omitted one or more entry items.

E7.1.1 Page server detects error and returns error messages to browser.

E7.1.2 Browser displays ‘login.html’ with active error messages.

E7.1.3 Return to step 5.

8. Page server sends SQL ‘SELECT’ statement for the ‘User’ table to the data server.

E8.1 Page server is unable to connect to data server or table.

E8.1.1 Page server receives an error 2002 (no such file or directory) or error 1146 (no such table)

9. Data server sends data to page server.

10. Page server receives data from data server and compares password from data server to entered password.

E10.1 User has entered an invalid email or a password that does not match.

E10.1.1 Page server detects error and returns error messages to browser.

E10.1.2 Browser displays ‘signIn.html’ with active error messages.

E10.1.3 Return to step 5.

11. Page server sends SQL ‘SELECT’ statement for the ‘Vehicle’ table to data server.

E11.1 Page server is unable to connect to data server or table.

E11.1.1 Page server receives an error 2002 (no such file or directory) or error 1146 (no such table)

12. Data server returns data to page server.

13. Page server updates and sends ‘home.html’ and registers the login to browser for display.

14. Browser displays ‘home.html’ in main section and signout in nav bar.

Post condition: User may continue to use case RM 1.4

**Case number:** RM 1.4

**Title (Goal):** View Movies

**Description:** User views a movies available to rent

**Primary Actors:** User, Browser, Page server, Data server

**Main:** Pre-condition: User is logged in and on the main page.

1. User clicks Movies’ button.

2. Browser sends request for ‘movies.html’ to page server.

3. Page server receives request and sends SQL ‘SELECT’ statement for the ‘Movies’ tables to data server.

E3.1 Page server is unable to locate ‘movies.html’.

E3.1.1 Page server sends ‘404 error’ to browser.

E3.2 Page server unable to connect to data server or table.

E3.2.1 Page server receives an error 2002 (no such file or directory) or error 1146 (no such table)

4. Data server sends data to page server.

5. Page server updates ‘movies.html’ and sends updated page to browser.

6. Browser receives and displays ‘movies.html’ in main section of template page with appropriate data in fields.

Post condition: User may continue to use case RM 1.5

**Case number:** RM 1.5

**Title (Goal):** Rent Movie

**Description:** User rents movie from their account.

**Primary Actors:** User, Browser, Page server, Data server

**Main:** Pre-condition: User is logged in and on the movies page.

1. User clicks ‘Cart’ button.

2. Browser sends request to the page server for cart.html’.

3. Page server receives request and sends ‘cart.html’.

E3.1 Page server is unable to locate ‘cart.html’.

E3.1.1 Page server sends ‘404 error’ to browser.

4. Browser receives and displays cart.html’ in main section of template page with appropriate data in fields.

5. User checks movie shown is the move they want to rent. The user then clicks the ‘Rent’ button.

6. Browser posts data to page server.

7. Page server receives and verifies data.

E7.1 Page server detects error and returns error messages to browser.

E7.1.1 Browser displays ‘cart.html’ with active error messages.

E7.1.2 Return to step 5.

8. Page server sends SQL ‘UPDATE’ statement for the ‘Movie’ table to data server.

E8.1 Page server is unable to connect to data server or table.

E8.1.1 Page server receives an error 2002 (no such file or directory) or error 1146 (no such table)

9. Page server sends SQL ‘SELECT’ statement for the ‘Movie’ table to data server.

E9.1 Page server is unable to connect to data server or table.

E9.1.1 Page server receives an error 2002 (no such file or directory) or error 1146 (no such table)

10. Page server updates and sends home.html’ to browser for display.

11. Browser displays home.html’ in main section of template page.

Post condition: User may continue to use cases: RM 1.1 – Rm 1.6

**Case number:** RM 1.6

**Title (Goal):** Movie History

**Description:** User views movies previously rented

**Primary Actors:** User, Browser, Page server, Data server

**Main:** Pre-condition: User is logged in and on the main page.

1. User clicks ‘History’ button.

2. Browser sends request for ‘history.html’ to page server.

3. Page server receives request and sends SQL ‘SELECT’ statement for the ‘Movies’ tables to data server.

E3.1 Page server is unable to locate ‘history.html’.

E3.1.1 Page server sends ‘404 error’ to browser.

E3.2 Page server unable to connect to data server or table.

E3.2.1 Page server receives an error 2002 (no such file or directory) or error 1146 (no such table)

4. Data server sends data to page server.

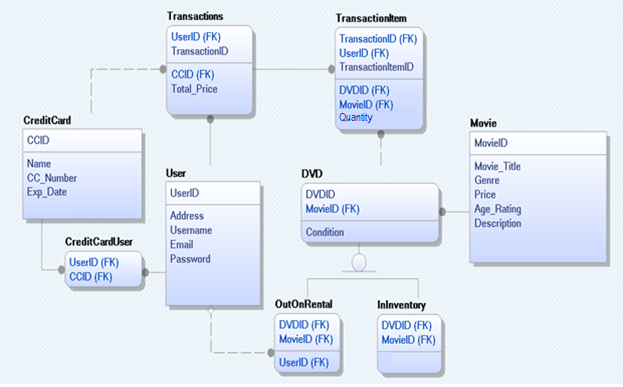
5. Page server updates ‘history.html’ and sends updated page to browser.

6. Browser receives and displays ‘history.html’ in main section of template page with appropriate data in fields.

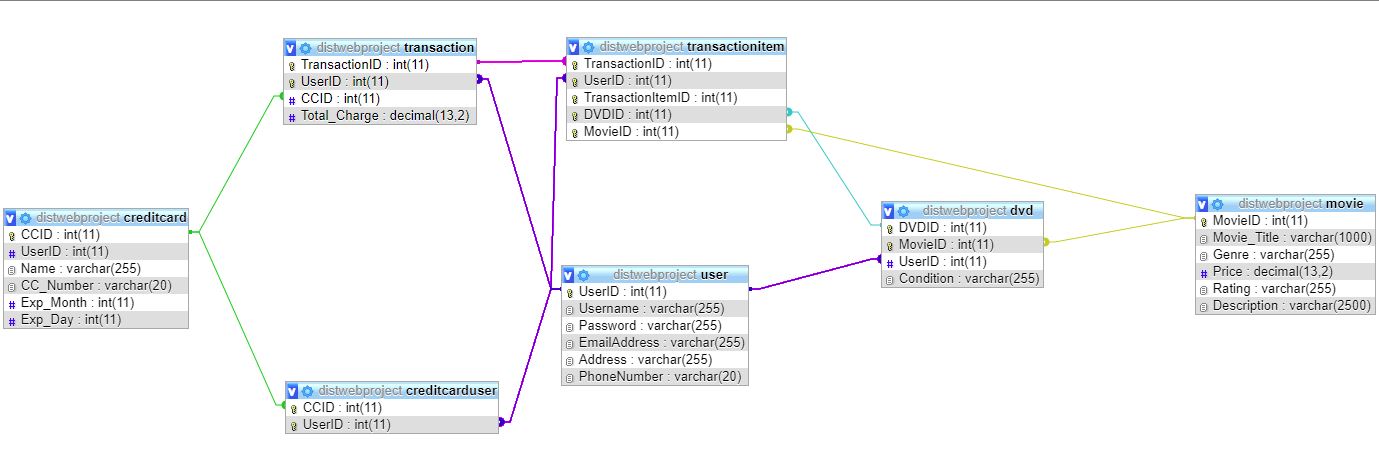
Post condition: User may continue to use case RM 1.1 - RM 1.6

**Diagrams**

**ER Diagram**

****

**Database Diagram**



**Referential Integrity Rules**

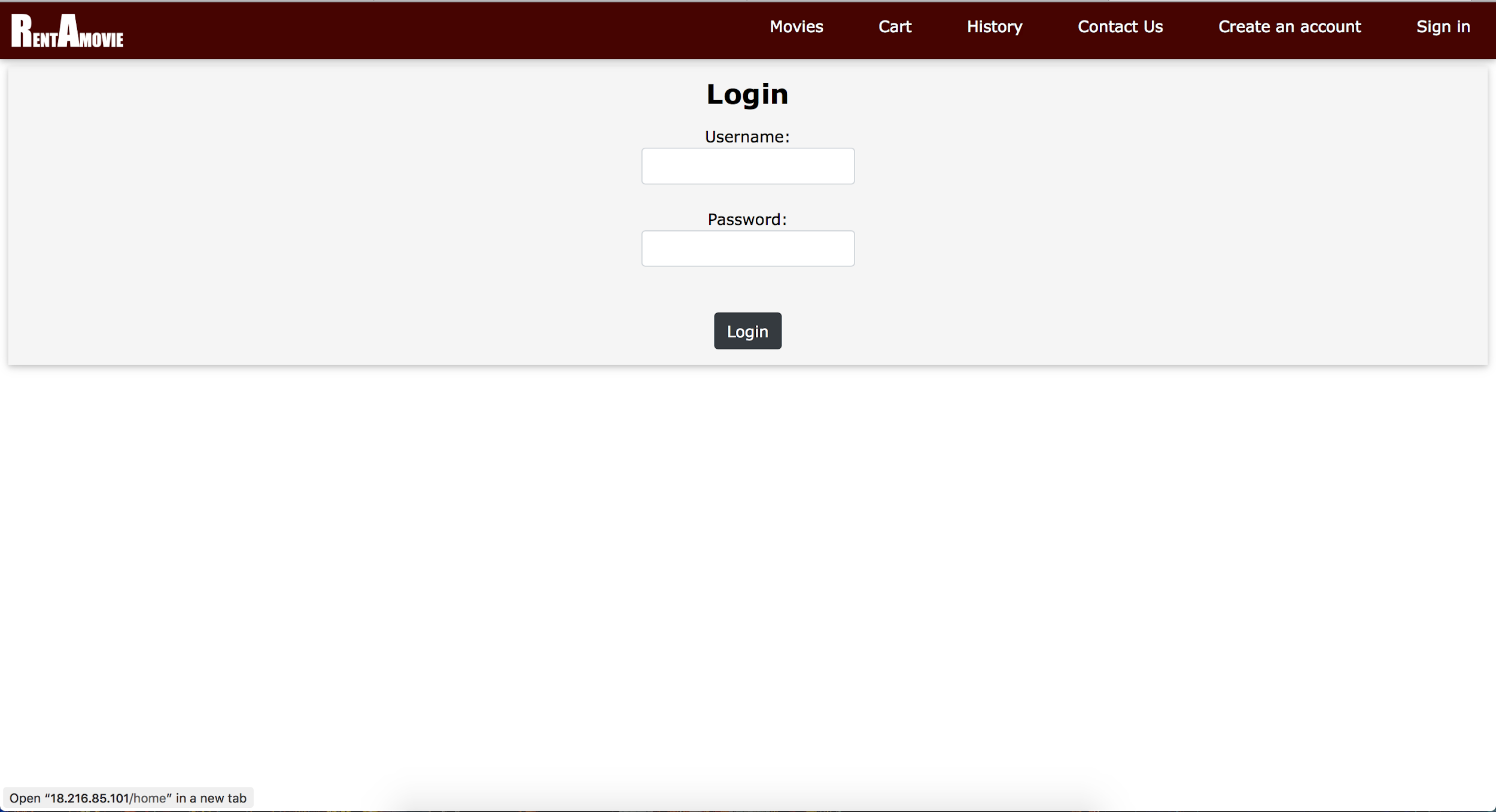
## 

|  |  |  |  |
| --- | --- | --- | --- |
| **Parent** | **Child** | **Action** | **Justification** |
| CreditCard | Transactions | Set Null | A transaction may still exist in the database if a credit card is removed; user should not have an order deleted because they deleted a credit card. |
| User | Transactions | Cascade | UserID is a part of a transaction’s primary key, so they cannot exist without the user. User accounts must be allowed to be removed because the website should not contain personal information without the user’s consent. |
| Transaction | TransactionItem | Cascade | Deleting a transaction needs to be allowed, and transactionItems are meaningless without an existing parent. |
| DVD | TransactionItem | Restrict | Instead of being removed from the database, retired DVDs should have a status indicating that they are retired. TransactionItems need to stay so users can still see their transaction history. |
| Movie | DVD | Restrict | Instead of being removed from the database, retired DVDs should have a status indicating that they are retired. TransactionItems need to stay so users can still see their transaction history. |
| User | DVD | Restrict | If a UserID is present in a dvd, that means it is currently out on rent to that user. Users should only be allowed to be removed if they are not currently renting a DVD. |
| User | CreditCardUser/CreditCard | Complex | If we remove a user, we would only remove a credit card from the CreditCard table if the card is not associated with another user. Obviously, the entry in the intermediate table would be deleted. |

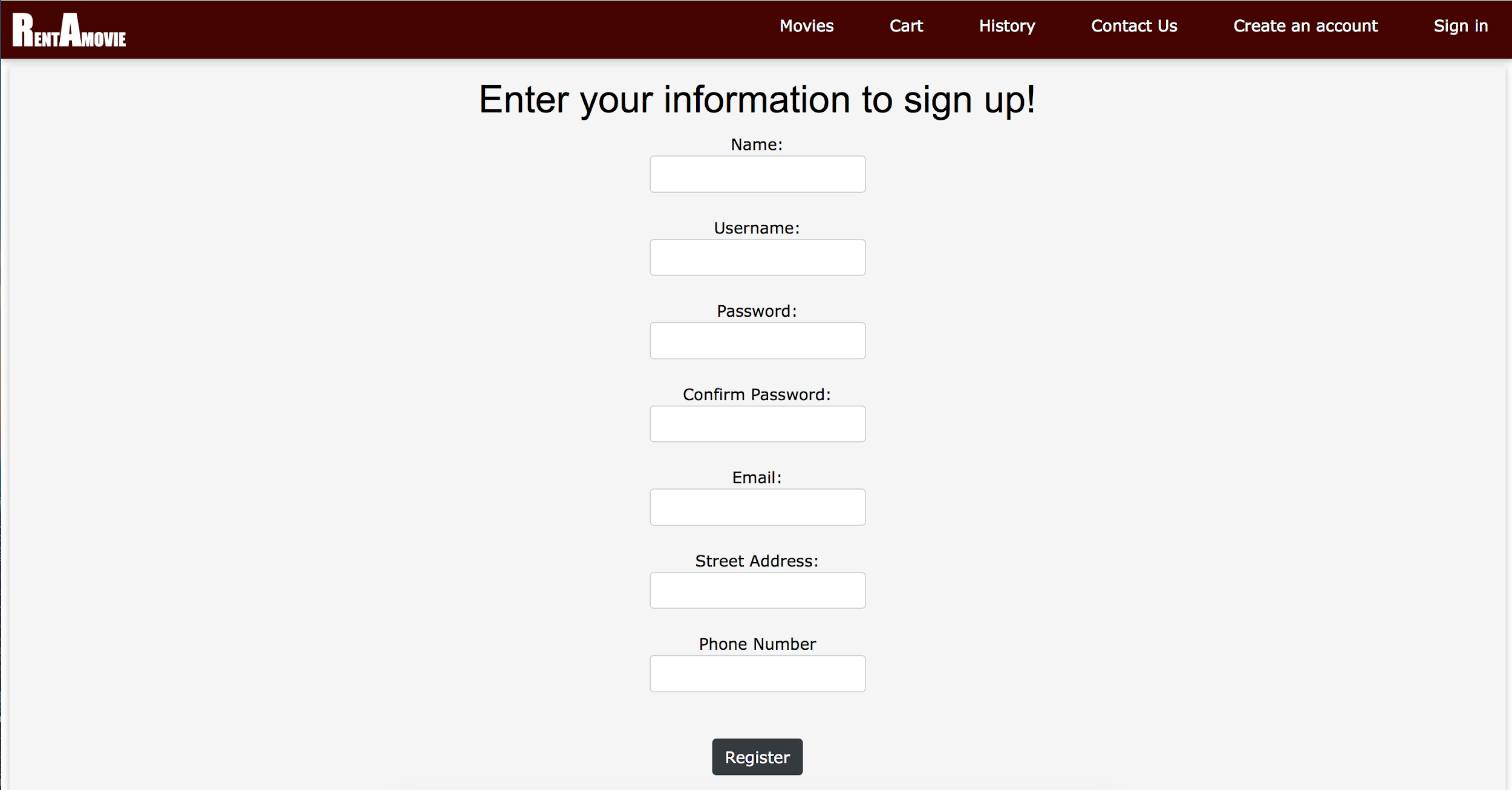
## 

**Website Designs**

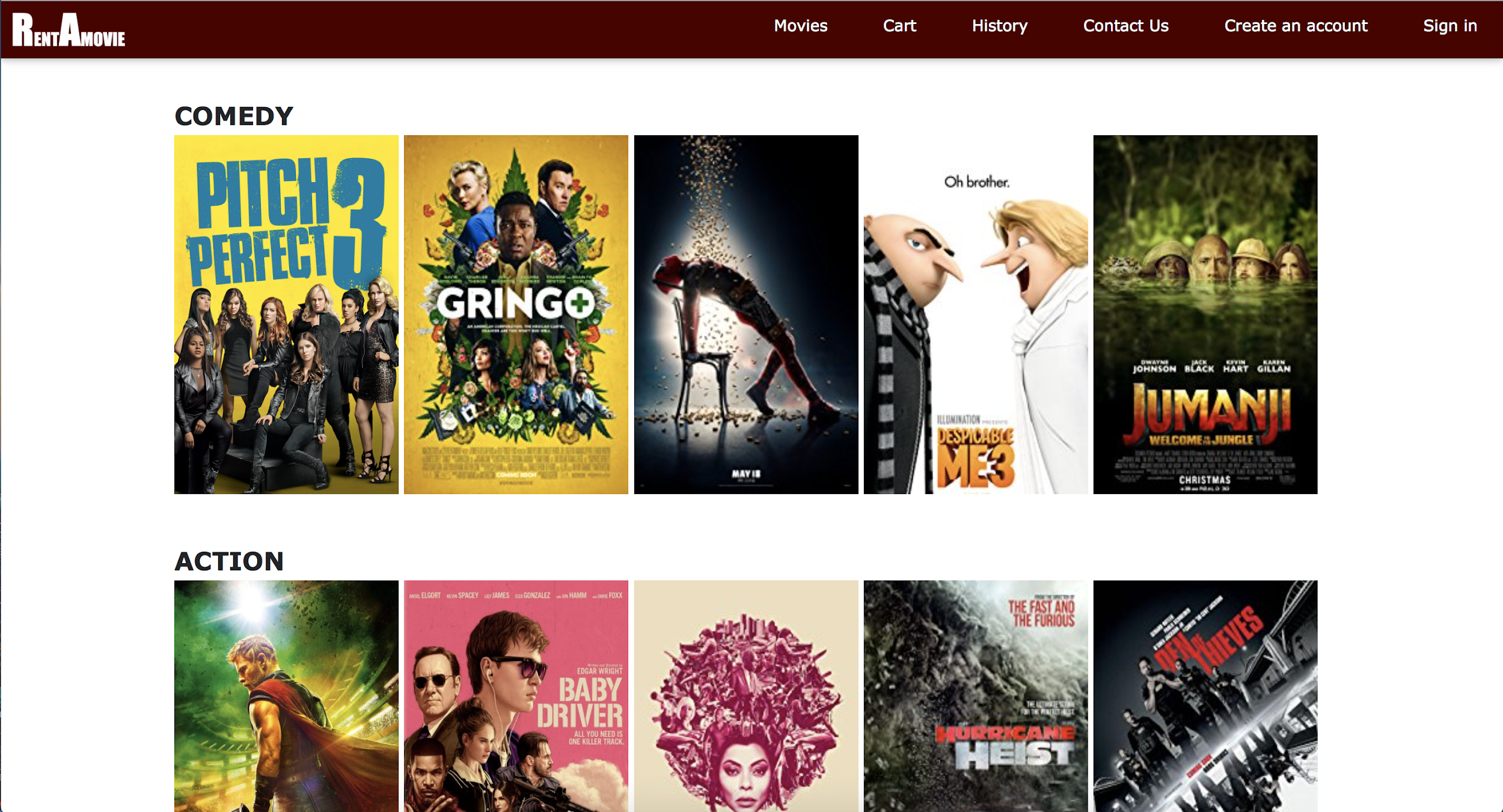
**Login page**

****

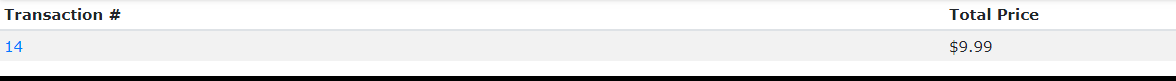
**Create Account Page**

****

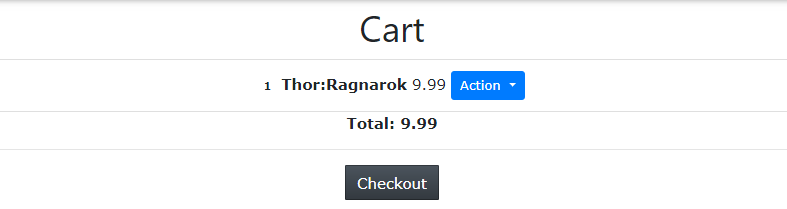
**Movies Page**

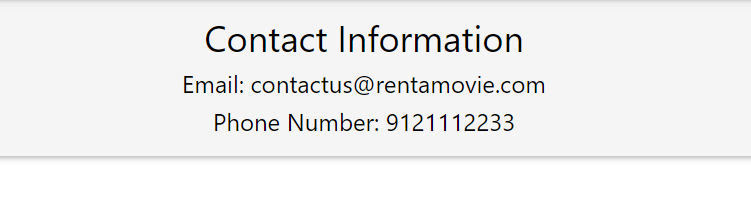
****

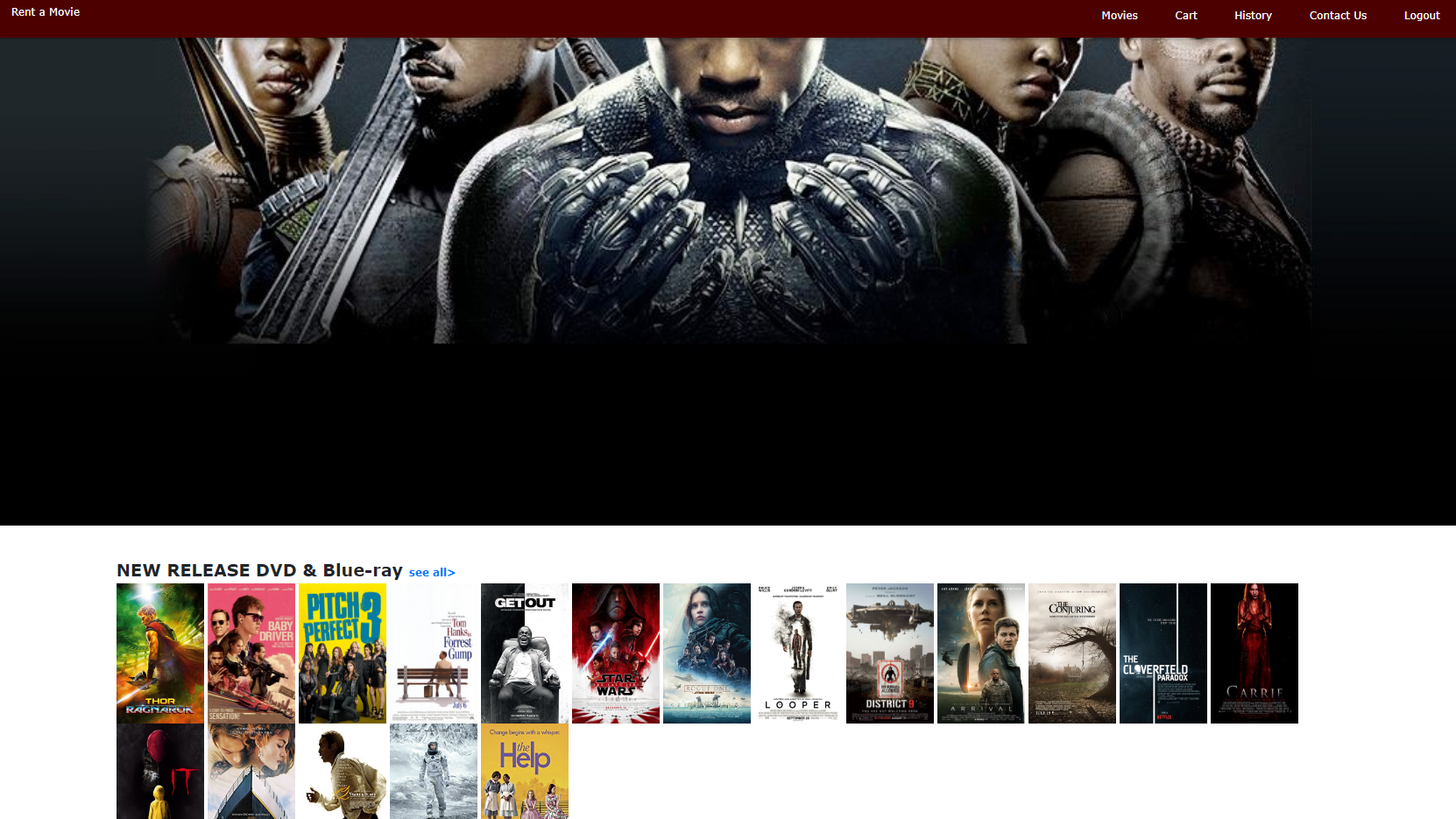
**History Page**



**Cart Page**

****

**Contact Page**

**Home Page**

**HTML Code**

**Login page**

**<!DOCTYPE html>  
<html>  
 <head>  
 <link rel="stylesheet" href="home.css">  
 </head>  
 <body>  
 <header class="w3-container w3-xlarge w3-padding-24" style="background-color: #0D10D4;">  
 <div class="w3-container">  
 <a href="login.html"><button class="w3-button w3-large w3-right" style="background-color: #0D10D4; color: #ffffff;">Login</button></a>  
 <a href="createAccount.html"><button class="w3-button w3-large w3-right" style="background-color: #0D10D4; color: #ffffff;">Create Account</button></a>  
 </div>  
 </header>  
 <div class="w3-bar w3-card" style="background-color: #8C8EFA; color: #ffffff;">  
 <a href="home.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Home</a>  
 <a href="movies.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Movies</a>  
 <a href="cart.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Cart</a>  
 <a href="history.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">History</a>  
 <a href="contact.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Contact Us</a>  
 </div>  
 <div class="w3-row-padding w3-center w3-margin-top">  
 <div class="w3-card w3-container" style="min-height: 150px; background-color: #8C8EFA; color: #000000;">  
 <h1>Login</h1>  
 <form action="">  
 Username:<br>  
 <input type="text" name="username">  
 <br>  
 Password:<br>  
 <input type="password" name="password">  
 <br><br>  
 <input type="submit" value="Submit">  
 </form>  
 </div>  
 </div>  
 </body>  
</html>**

**Create Account Page**

**<!DOCTYPE html>  
<html>  
 <head>  
 <link rel="stylesheet" href="home.css">  
 </head>  
 <body>  
 <header class="w3-container w3-xlarge w3-padding-24" style="background-color: #0D10D4;">  
 <div class="w3-container">  
 <a href="login.html"><button class="w3-button w3-large w3-right" style="background-color: #0D10D4; color: #ffffff;">Login</button></a>  
 <a href="createAccount.html"><button class="w3-button w3-large w3-right" style="background-color: #0D10D4; color: #ffffff;">Create Account</button></a>  
 </div>  
 </header>  
 <div class="w3-bar w3-card" style="background-color: #8C8EFA; color: #ffffff;">  
 <a href="home.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Home</a>  
 <a href="movies.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Movies</a>  
 <a href="cart.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Cart</a>  
 <a href="history.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">History</a>  
 <a href="contact.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Contact Us</a>  
 </div>  
 <div class="w3-row-padding w3-center w3-margin-top">  
 <div class="w3-card w3-container" style="min-height: 150px; background-color: #8C8EFA; color: #000000;">  
 <h1>Enter your information to sign up!</h1>  
 <form>  
 Username:<br>  
 <input type="text" name="username"><br>  
 Password:<br>  
 <input type="password" name="password"><br>  
 Email:<br>  
 <input type="email" name="email"><br>  
 Street Address:<br>  
 <input type="text" name="address"><br>  
 City:<br>  
 <input type="text" name="city"><br>  
 State:<br>  
 <input type="text" name="state"><br>  
 ZIP Code:<br>  
 <input type="number" name="zipcode"><br>  
 Phone Number<br>  
 <input type="tel" name="phone"><br>  
 <br><input type="submit" value="Submit">  
 </form>  
 </div>  
 </div>  
 </body>  
</html>**

**History Page**

**<!DOCTYPE html>  
<html>  
 <head>  
 <link rel="stylesheet" href="home.css">  
 </head>  
 <body>  
 <header class="w3-container w3-xlarge w3-padding-24" style="background-color: #0D10D4;">  
 <div class="w3-container">  
 <a href="login.html"><button class="w3-button w3-large w3-right" style="background-color: #0D10D4; color: #ffffff;">Login</button></a>  
 <a href="createAccount.html"><button class="w3-button w3-large w3-right" style="background-color: #0D10D4; color: #ffffff;">Create Account</button></a>  
 </div>  
 </header>  
 <div class="w3-bar w3-card" style="background-color: #8C8EFA; color: #ffffff;">  
 <a href="home.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Home</a>  
 <a href="movies.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Movies</a>  
 <a href="cart.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Cart</a>  
 <a href="history.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">History</a>  
 <a href="contact.html" class="w3-bar-item w3-button w3-padding-large w3-hover-white">Contact Us</a>  
 </div>  
 <div class="w3-row-padding w3-center w3-margin-top">  
 <div class="w3-card w3-container" style="min-height: 150px; background-color: #8C8EFA; color: #000000;">  
 <h1>Movies you have watched</h1>  
 <?php  
 ?>  
 </div>  
 </div>  
 </body>  
</html>**

**Database Code/Queries**

**Database Script**

**SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";**

**SET AUTOCOMMIT = 0;**

**START TRANSACTION;**

**SET time\_zone = "+00:00";**

**/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;**

**/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;**

**/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;**

**/\*!40101 SET NAMES utf8mb4 \*/;**

**--**

**-- Database: `distwebproject`**

**--**

**-- --------------------------------------------------------**

**--**

**-- Table structure for table `creditcard`**

**--**

**CREATE TABLE `creditcard` (**

**`CCID` int(11) NOT NULL,**

**`UserID` int(11) NOT NULL,**

**`Name` varchar(255) NOT NULL,**

**`CC\_Number` varchar(20) NOT NULL,**

**`Exp\_Month` int(11) NOT NULL,**

**`Exp\_Day` int(11) NOT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**-- --------------------------------------------------------**

**--**

**-- Table structure for table `creditcarduser`**

**--**

**CREATE TABLE `creditcarduser` (**

**`CCID` int(11) NOT NULL,**

**`UserID` int(11) NOT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**-- --------------------------------------------------------**

**--**

**-- Table structure for table `dvd`**

**--**

**CREATE TABLE `dvd` (**

**`DVDID` int(11) NOT NULL,**

**`MovieID` int(11) NOT NULL,**

**`UserID` int(11) DEFAULT NULL COMMENT 'Is null for DVDs not out on rent.',**

**`Condition` varchar(255) NOT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**-- --------------------------------------------------------**

**--**

**-- Table structure for table `movie`**

**--**

**CREATE TABLE `movie` (**

**`MovieID` int(11) NOT NULL,**

**`Movie\_Title` varchar(1000) NOT NULL COMMENT '1000 char limit for absurdly long movie titles.',**

**`Genre` varchar(255) NOT NULL COMMENT 'Each movie may only have 1 genre for now.',**

**`Price` decimal(13,2) NOT NULL,**

**`Rating` varchar(255) DEFAULT NULL,**

**`Description` varchar(2500) DEFAULT NULL COMMENT '2500 character limit for the movie description'**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**-- --------------------------------------------------------**

**--**

**-- Table structure for table `transaction`**

**--**

**CREATE TABLE `transaction` (**

**`TransactionID` int(11) NOT NULL,**

**`UserID` int(11) NOT NULL,**

**`CCID` int(11) NOT NULL,**

**`Total\_Charge` decimal(13,2) NOT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**-- --------------------------------------------------------**

**--**

**-- Table structure for table `transactionitem`**

**--**

**CREATE TABLE `transactionitem` (**

**`TransactionID` int(11) NOT NULL,**

**`UserID` int(11) NOT NULL,**

**`TransactionItemID` int(11) NOT NULL,**

**`DVDID` int(11) NOT NULL,**

**`MovieID` int(11) NOT NULL,**

**`Quantity` int(11) NOT NULL**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**-- --------------------------------------------------------**

**--**

**-- Table structure for table `user`**

**--**

**CREATE TABLE `user` (**

**`UserID` int(11) NOT NULL,**

**`Username` varchar(255) NOT NULL,**

**`Password` varchar(255) NOT NULL,**

**`EmailAddress` varchar(255) NOT NULL COMMENT 'User Email',**

**`Address` varchar(255) NOT NULL COMMENT 'User Address',**

**`PhoneNumber` varchar(20) NOT NULL COMMENT 'User Phone Number'**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**--**

**-- Indexes for dumped tables**

**--**

**--**

**-- Indexes for table `creditcard`**

**--**

**ALTER TABLE `creditcard`**

**ADD PRIMARY KEY (`CCID`);**

**--**

**-- Indexes for table `creditcarduser`**

**--**

**ALTER TABLE `creditcarduser`**

**ADD PRIMARY KEY (`CCID`,`UserID`),**

**ADD KEY `UserID` (`UserID`);**

**--**

**-- Indexes for table `dvd`**

**--**

**ALTER TABLE `dvd`**

**ADD PRIMARY KEY (`DVDID`,`MovieID`),**

**ADD KEY `MovieID` (`MovieID`),**

**ADD KEY `UserID` (`UserID`);**

**--**

**-- Indexes for table `movie`**

**--**

**ALTER TABLE `movie`**

**ADD PRIMARY KEY (`MovieID`);**

**--**

**-- Indexes for table `transaction`**

**--**

**ALTER TABLE `transaction`**

**ADD PRIMARY KEY (`TransactionID`,`UserID`),**

**ADD KEY `UserID` (`UserID`),**

**ADD KEY `CCID` (`CCID`);**

**--**

**-- Indexes for table `transactionitem`**

**--**

**ALTER TABLE `transactionitem`**

**ADD PRIMARY KEY (`TransactionItemID`),**

**ADD UNIQUE KEY `TransactionID` (`TransactionID`),**

**ADD UNIQUE KEY `UserID` (`UserID`),**

**ADD UNIQUE KEY `DVDID` (`DVDID`),**

**ADD UNIQUE KEY `MovieID` (`MovieID`);**

**--**

**-- Indexes for table `user`**

**--**

**ALTER TABLE `user`**

**ADD PRIMARY KEY (`UserID`);**

**--**

**-- Constraints for dumped tables**

**--**

**--**

**-- Constraints for table `creditcarduser`**

**--**

**ALTER TABLE `creditcarduser`**

**ADD CONSTRAINT `creditcarduser\_ibfk\_1` FOREIGN KEY (`CCID`) REFERENCES `creditcard` (`CCID`),**

**ADD CONSTRAINT `creditcarduser\_ibfk\_2` FOREIGN KEY (`UserID`) REFERENCES `user` (`UserID`);**

**--**

**-- Constraints for table `dvd`**

**--**

**ALTER TABLE `dvd`**

**ADD CONSTRAINT `dvd\_ibfk\_1` FOREIGN KEY (`MovieID`) REFERENCES `movie` (`MovieID`),**

**ADD CONSTRAINT `dvd\_ibfk\_2` FOREIGN KEY (`UserID`) REFERENCES `user` (`UserID`);**

**--**

**-- Constraints for table `transaction`**

**--**

**ALTER TABLE `transaction`**

**ADD CONSTRAINT `transaction\_ibfk\_1` FOREIGN KEY (`UserID`) REFERENCES `user` (`UserID`),**

**ADD CONSTRAINT `transaction\_ibfk\_2` FOREIGN KEY (`CCID`) REFERENCES `creditcard` (`CCID`);**

**--**

**-- Constraints for table `transactionitem`**

**--**

**ALTER TABLE `transactionitem`**

**ADD CONSTRAINT `transactionitem\_ibfk\_1` FOREIGN KEY (`UserID`) REFERENCES `user` (`UserID`) ON DELETE CASCADE ON UPDATE CASCADE,**

**ADD CONSTRAINT `transactionitem\_ibfk\_2` FOREIGN KEY (`TransactionID`) REFERENCES `transaction` (`TransactionID`) ON DELETE CASCADE ON UPDATE CASCADE,**

**ADD CONSTRAINT `transactionitem\_ibfk\_3` FOREIGN KEY (`DVDID`) REFERENCES `dvd` (`DVDID`) ON DELETE CASCADE ON UPDATE CASCADE,**

**ADD CONSTRAINT `transactionitem\_ibfk\_4` FOREIGN KEY (`MovieID`) REFERENCES `movie` (`MovieID`) ON DELETE CASCADE ON UPDATE CASCADE;**

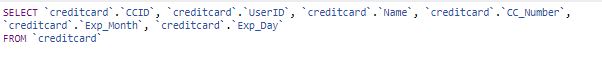
**COMMIT;**

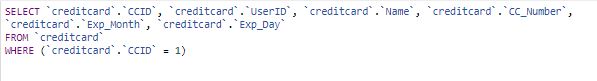
**/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;**

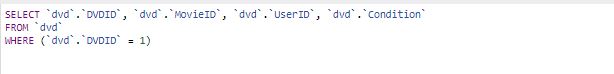
**/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;**

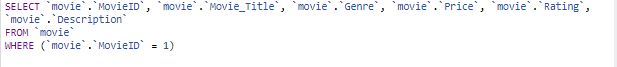
**/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;**

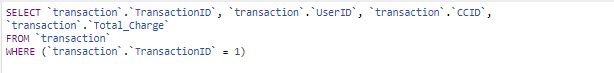
**Queries**

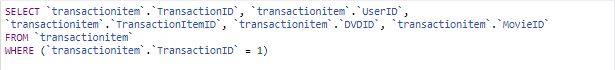
****

****

****

****

****

****

****

**PHP**

**HomeController.php**

**<?php**

**namespace App\Http\Controllers;**

**use Illuminate\Http\Request;**

**use App\Movie;**

**class HomeController extends Controller**

**{**

**public function index() {**

**if(!auth()->check())**

**return redirect()->to('/login');**

**$movies = (new Movie)->getHomeMovies();**

**$bp = (new Movie)->movie('4');**

**return view('home.index', compact('movies', 'bp'));**

**}**

**public function contact() {**

**return view('contact.contact');**

**}**

**}**

**MovieController.php**

**<?php**

**namespace App\Http\Controllers;**

**use Illuminate\Http\Request;**

**use App\Movie;**

**class MoviesController extends Controller**

**{**

**public function index() {**

**$actions = (new Movie)->getMovie('action');**

**$comedies = (new Movie)->getMovie('comedy');**

**$dramas = (new Movie)->getMovie('drama');**

**$horrors = (new Movie)->getMovie('horror');**

**$scifis = (new Movie)->getMovie('sci-fi');**

**return view('movies.index', compact('actions', 'comedies', 'dramas', 'horrors', 'scifis'));**

**}**

**public function description(Movie $movie) {**

**return view('movies.description', compact('movie'));**

**}**

**}**

**RegisterController.php**

**<?php**

**namespace App\Http\Controllers;**

**use App\User;**

**use Illuminate\Support\Facades\DB;**

**use Illuminate\Http\Request;**

**class RegisterController extends Controller**

**{**

**public function index() {**

**return view('register.index');**

**}**

**public function store() {**

**$this->validate(request(), [**

**'name' => 'required',**

**'username' => 'required|unique:users',**

**'email' => 'required|email|unique:users',**

**'password' => 'required|regex:/^.\*(?=.{3,})(?=.\*[a-zA-Z])(?=.\*[0-9])(?=.\*[\d\X])(?=.\*[!$#%]).\*$/|confirmed',**

**'address' => 'required',**

**'phone' => 'required',**

**]);**

**$user = User::create(request(['name', 'username', 'email', 'password', 'address', 'phone']));**

**auth()->login($user);**

**return redirect()->to('/home');**

**}**

**}**

**SessionsController.php**

**<?php**

**namespace App\Http\Controllers;**

**use Illuminate\Http\Request;**

**class SessionsController extends Controller**

**{**

**public function index() {**

**return view('sessions.index');**

**}**

**public function store() {**

**if(auth()->attempt(request(['username', 'password'])) == false) {**

**return back()->withErrors([**

**'message' => 'The username or password is incorrect, please try again'**

**]);**

**}**

**return redirect()->to('/home');**

**}**

**public function destroy() {**

**auth()->logout();**

**session()->flush();**

**return redirect()->to('/');**

**}**

**}**

**ShopController.php**

**<?php**

**namespace App\Http\Controllers;**

**use App\Movie;**

**use App\Cart;**

**use Illuminate\Http\Request;**

**use Illuminate\Support\Facades\DB;**

**use Session;**

**class ShopController extends Controller{**

**public function index() {**

**if(!auth()->check())**

**return redirect()->to('/login');**

**return view('shop.index');**

**}**

**public function getReduceByOne($id) {**

**$movie = (new Movie)->find($id);**

**$oldcart = Session::has('cart') ? Session::get('cart') : null;**

**$cart = new Cart($oldcart);**

**$cart->reduceByOne($movie, $id);**

**session()->put('cart', $cart);**

**return redirect()->to('/cart');**

**}**

**public function getRemoveItem($id) {**

**$oldcart = Session::has('cart') ? Session::get('cart') : null;**

**$cart = new Cart($oldcart);**

**$cart->removeItem($id);**

**session()->put('cart', $cart);**

**return redirect()->to('/cart');**

**}**

**public function store(Request $request) {**

**$movie = (new Movie)->find($request->input('id'));**

**$oldcart = Session::has('cart') ? Session::get('cart') : null;**

**$cart = new Cart($oldcart);**

**$cart->add($movie, $movie->id);**

**$request->session()->put('cart', $cart);**

**return redirect()->to('/movies');**

**}**

**public function getCart() {**

**if (!auth()->check())**

**return redirect()->to('/login');**

**elseif (!Session::has('cart'))**

**return view('shop.index');**

**$oldcart = Session::get('cart');**

**$cart = new Cart($oldcart);**

**return view('shop.index', [**

**'movies' => $cart->movies,**

**'totalPrice' => $cart->totalPrice,**

**]);**

**}**

**public function getCheckout() {**

**if(!Session::has('cart')) {**

**return view('shop.index');**

**}**

**$oldcart = Session::get('cart');**

**$cart = new Cart($oldcart);**

**return view('shop.checkout', [**

**'movies' => $cart->movies,**

**'total' => $cart->totalPrice,**

**]);**

**}**

**public function buy(Request $request) {**

**if(!Session::has('cart')) {**

**return view('shop.index');**

**}**

**$id = \Auth::id();**

**$oldcart = session()->pull('cart');**

**$cart = new Cart($oldcart);**

**$transactionid = DB::table('transaction')->insertGetId([**

**'UserID' => $id,**

**'CCID' => '1',**

**'Total\_Charge' => $cart->totalPrice**

**]);**

**foreach ($cart->movies as $movie)**

**DB::table('transactionitem')->insert([**

**'TransactionID' => $transactionid,**

**'UserID' => $id,**

**'MovieID' => $movie['item']['id']**

**]);**

**$ccid = DB::table('creditcard')->insertGetId([**

**'UserID' => $id,**

**'Name' => $request->input('card-holder'),**

**'CC\_Number' => $request->input('card-number'),**

**'Exp\_Month' => $request->input('MM'),**

**'Exp\_Day' => $request->input('YY')**

**]);**

**DB::table('creditcarduser')->insert([**

**'CCID' => $ccid,**

**'UserID' => $id**

**]);**

**return view('shop.index')->with('success', 'Purchase is Successful');**

**}**

**public function history() {**

**if(!auth()->check())**

**return redirect()->to('/login');**

**$id = \Auth::id();**

**$transactions = DB::table('transaction')->where('UserID', '=', $id)->get();**

**return view('history.index', compact('transactions'));**

**}**

**public function transaction($transaction) {**

**if(!auth()->check())**

**return redirect()->to('/login');**

**$id = \Auth::id();**

**$vid = DB::table('transaction')->where('TransactionID', '=', $transaction)->pluck('UserID')->first();**

**if($id != $vid)**

**return redirect()->to('/home');**

**$mids = DB::table('transactionitem')->where('TransactionID', '=', $transaction)->pluck('MovieID');**

**$movies = array();**

**$i = 0;**

**foreach ($mids as $mid) {**

**$movies[$i] = (new Movie)->getHistory($mid);**

**$i++;**

**}**

**return view('history.transaction', compact('movies'));**

**}**

**}**

**Cart.php**

**<?php**

***/\*\****

***\* Created by PhpStorm.***

***\* User: Mike***

***\* Date: 4/22/18***

***\* Time: 12:35 AM***

***\*/***

**namespace App;**

**class Cart**

**{**

**public $movies = null, $totalQty = 0, $totalPrice = 0;**

**public function \_\_construct($oldcart)**

**{**

**if ($oldcart) {**

**$this->movies = $oldcart->movies;**

**$this->totalQty = $oldcart->totalQty;**

**$this->totalPrice = $oldcart->totalPrice;**

**}**

**}**

**public function add($movie, $id) {**

**$storedItem = [**

**'qty' => 0,**

**'price' => $movie->Price,**

**'item' => $movie,**

**];**

**if($this->movies) {**

**if(array\_key\_exists($id, $this->movies)) {**

**$storedItem = $this->movies[$id];**

**}**

**}**

**$storedItem['qty']++;**

**$storedItem['price'] = $movie->Price \* $storedItem['qty'];**

**$this->movies[$id] = $storedItem;**

**$this->totalQty++;**

**$this->totalPrice += $movie->Price;**

**}**

**public function reduceByOne($movie, $id) {**

**$this->movies[$id]['qty']--;**

**$this->movies[$id]['price'] -= $movie->Price;**

**$this->totalQty--;**

**$this->totalPrice -= $movie->Price;**

**if($this->movies[$id]['qty'] <= 0)**

**unset($this->movies[$id]);**

**}**

**public function removeItem($id) {**

**$this->totalQty -= $this->movies[$id]['qty'];**

**$this->totalPrice -= $this->movies[$id]['price'];**

**unset($this->movies[$id]);**

**}**

**}**

**Movie.php**

**<?php**

**namespace App;**

**use Illuminate\Database\Eloquent\Model;**

**class Movie extends Model**

**{**

**public function getMovie($genre) {**

**return $this->where('genre', $genre)->where('id', '!=', '4')->get();**

**}**

**public function getHomeMovies() {**

**return $this->where('id', '!=', '4')->take(10)->get();**

**}**

**public function getMovies() {**

**return $this->where('id', '!=', '4')->get();**

**}**

**public function getHistory($id) {**

**return $this->where('id', $id)->get();**

**}**

**public function scopeMovie($query, $id) {**

**return $query->find($id);**

**}**

**public function scopeInfo($query, $info) {**

**return $query->pluck($info);**

**}**

**public function user() {**

**$this->belongsTo(User::class);**

**}**

**}**

**User.php**

**<?php**

**namespace App;**

**use Illuminate\Notifications\Notifiable;**

**use Illuminate\Foundation\Auth\User as Authenticatable;**

**class User extends Authenticatable**

**{**

**use Notifiable;**

***/\*\****

***\* The attributes that are mass assignable.***

***\****

***\* @var array***

***\*/***

**protected $fillable = [**

**'name', 'username', 'email', 'password', 'phone', 'address',**

**];**

***/\*\****

***\* The attributes that should be hidden for arrays.***

***\****

***\* @var array***

***\*/***

**protected $hidden = [**

**'password', 'remember\_token',**

**];**

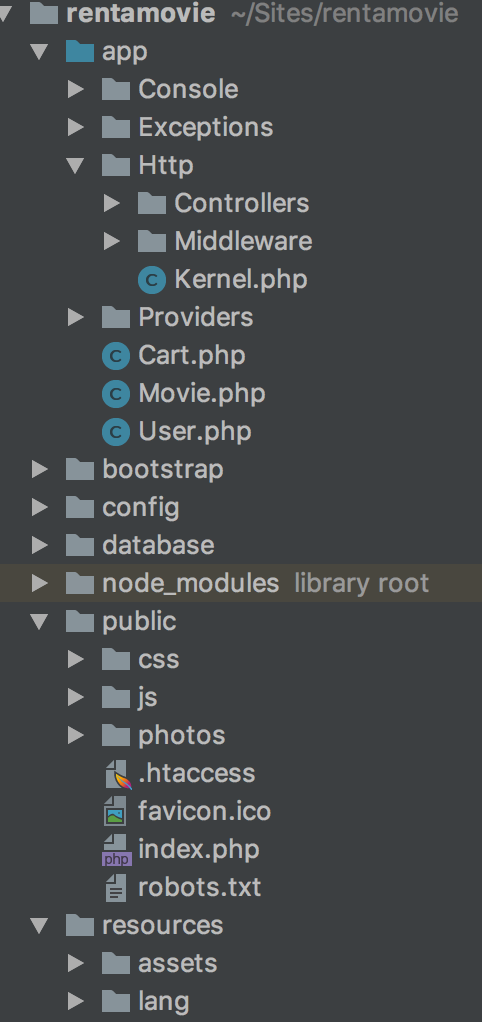
**public function setPasswordAttribute($password) {**

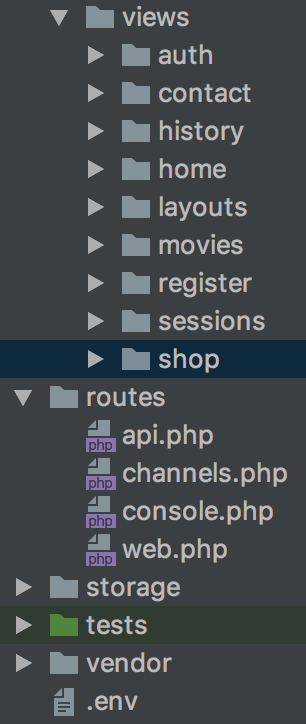
**$this->attributes['password'] = bcrypt($password);**

**}**

**}**

**Site Structure**





**Integration Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| Test #: 1 | | | Test Name: Sign Up |
| Description: User goes to “Create an account” and input require information. | | | |
| Step# | Description | Pass/Fail | |
| 1 | Navigate to “Create an account”. | Pass | |
| 2 | Input information required. | Pass | |
| 3 | Click “Register” button. | Pass | |
| Post-Conditions:  User’s data is added to the database and the user is redirected to the main page. | | | |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Test #: 2 | | | Test Name: Sign In |
| Description: User inputs their credentials and clicks the “Sign In” button to be directed to the main page. | | | |
| Step# | Description | Pass/Fail | |
| 1 | Input required sign in information. | Pass | |
| 2 | Click “Sign In” button. | Pass | |
| Post-Conditions:  User’s input data is compared with the information in the database and redirected to the main page if credentials match. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test #: 3 | | | Test Name: Add to Cart |
| Description: User clicks on desired movie and adds it to their cart. | | | |
| Step# | Description | Pass/Fail | |
| 1 | Navigate to desired movie. | Pass | |
| 2 | Click on movie. | Pass | |
| 3 | Click “Add to Cart” button. | Pass | |
| 4 | Navigate to cart by clicking “Cart” button at top right. | Pass | |
| Post-Conditions:  User’s desired movie is now in the cart and ready to be purchased. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test #: 4 | | | Test Name: Checkout |
| Description: User adds desired movies to “Cart” and clicks checkout. User then clicks “Checkout”, inputs credit card information then clicks “Proceed” to purchase. | | | |
| Step# | Description | Pass/Fail | |
| 1 | Navigate and adds desired movie to cart. | Pass | |
| 2 | Click on “Cart”. | Pass | |
| 3 | Click “Checkout”. | Pass | |
| 4 | Input card information then click “Proceed”. | Pass | |
| Post-Conditions: User’s cart is now cleared and (theoretically) their card is charged and the movie is sent to the user. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test #: 5 | | | Test Name: Remove movie |
| Description: User removes unwanted movies from cart. | | | |
| Step# | Description | Pass/Fail | |
| 1 | Navigate to cart by clicking “Cart”. | Pass | |
| 2 | Click the “Action” button to display drop down box. | Pass | |
| 3 | Click “Remove 1” or “Remove All” button to remove movies. | Pass | |
| Post-Conditions: The unwanted movie is removed from the user’s cart. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test #: 6 | | | Test Name: History |
| Description: User reviews movies they have previously rented. | | | |
| Step# | Description | Pass/Fail | |
| 1 | Navigate to History by clicking “History”. | Pass | |
| 2 | Click the “Transaction Number” to display movies rented. | Pass | |
| 3 | Click movie title for movie information. | Pass | |
| Post-Conditions: The user can see what movies they have previously rented and how much they have spent on this site. | | | |

**Conclusion**

This project taught us the principles of creating a interactive website combining CSS,AJAX, PHP and SQL. As a group we learned about both the client and server side of web design. We also experienced collaborating as a team to plan, design and create this site. Each of us contributed equally to add features to the site and test it for any problems. Overall, our group was able to consolidate ideas, skills, and talents to complete this assignment successfully.