İhsan Doğramacı Bilkent University

Department of Computer Science

CS353 – DATABASE SYSTEMS

Online Movie Rental System

Final Report

Group 20:

Efe Ertürk 21902620

Oğuz Ata Çal 21903088

Yiğit Ekin 21901784

Arda Eren 21902505

1. Introduction

The topic of this project is designing an Online Movie Rental System. Our aim in this proposal is to clearly describe the project topic that we are designing and to discuss why and how we use a database for this system.

Then, we will specify functional and non-functional requirements for this system that detail the interaction between the different types of users and the system functionalities along with the system constraints. In the end of this section, we will list the programming language and frameworks we plan to use under the pseudo-requirements heading.

Finally, we will display the conceptual design of our database using an entity-relationship model that will include strong and weak entities, relationships between entities, cardinality constraints, keys and attributes and more.

2. Project Description

The main aim of this Online Movie Rental system is to provide customers a platform to easily rent movies online. Customers will be able to create a request for a movie to be added to the system if their desired movie is not available. The customer can also write reviews, rate films and like or dislike other reviews. While writing a review, one can warn users if the review includes a spoiler to the film or not. The reviews can be displayed according to their dates, from newest to oldest, or according to net likes, or one can choose to view the reviews that do not have any spoiler in it and so on. There is also a friends system that allows customers to add other customers as friends and recommend films to them. The recommendation can include a message for your friends too.

Customers will be able to search movies by title, genre, production year, director and actors. They will also be able to see the movies they are currently renting, their rental history showing the movies they rented previously, the movies that their friends recommended to them, the most rented movies, the best rated movies (which has the best average rating) and finally the list of movies that they put on their favorites. The system will also keep track of the twenty movies that are newly registered to the system.

Employees will be able to satisfy the requests of customer requests and register new movies to the system. They will also have the authority to delete customer accounts.

For the new customers, there will be a promotion code assigned to them, which enables them to rent films at a discounted price.

3. Project Contribution Summary

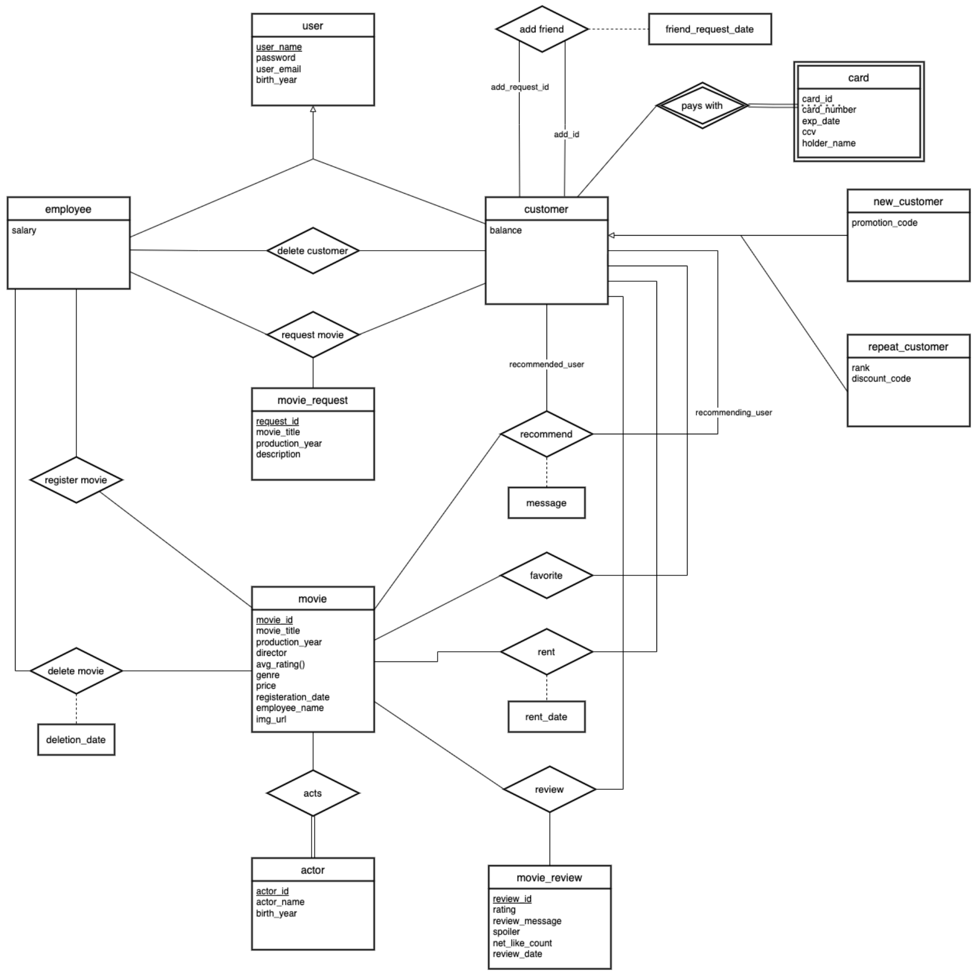
Yiğit Ekin: Yiğit was responsible for the front-end implementation of the project. He designed and implemented the UI. He also created the UI mockups for the design report. He and Oğuz connected the backend and frontend of the project and tested the application.

Efe Ertürk: Efe was one of the members of the backend group of this project. He and Oğuz implemented the functions and the database system of the project. He also helped write the reports.

Oğuz Ata Çal: Oğuz was the other member of the backend group. He and Efe implemented the functions and the database system of the project. He worked with Yiğit while connecting the frontend with the backend of the application and while testing. He also helped write the reports.

Arda Eren: Arda helped Yiğit in UI design and he did the code review. He wrote the reports and created the ER and relational models. He also helped the testing stage of the system.

4. Entity-Relationship Model



5. Relational Model

User(user_name, password, user_email, birth_year)

Employee(employee_name, salary)
employee_name foreign key to user_name of User

Customer(customer_name, balance)
customer_name foreign key to user_name of User

New_customer(customer_name, promotion_code)
customer_name foreign key to customer_name of Customer

Repeat_customer(customer_name, rank, discount_code)
customer_name foreign key to customer_name of Customer

Movie(movie_id, movie_title, production_year, director, avg_rating, genre, price,employee_name, registeration_date, img_url)

Movie_request(request_id, movie_title, production_year, description, customer_name)
customer_name foreign key to customer_name of Customer

Friend(add_request_name, add_name, friend_request_date)
add_request_name foreign key to customer_name of Customer
add_name foreign key to customer_name of Customer

Card(card_id,customer_name, card_number, exp_date, ccv, holder_name)

Actor(actor_id, actor_name, birth_year)

Acts(actor_id, movie_id)
actor_id foreign key to actor_id of Actor
movie_id foreign key to movie_id of Movie

Recommend(recommend_id, recomended_user_name, recommender_user_name, movie_id, message)
movie_id foreign key to movei_id of Movie
recommended_user_name foreign key to customer_name of Customer
recommender_user_name foreign key to customer_name of Customer

Favorite(customer_name, movie_id)
customer_name foreign key to customer_name of Customer
movie_id foreign key to movie_id of Movie

Rent(rent_id, customer_name, movie_id, rent_date)
customer_name foreign key to customer_name of Customer
movie_id foreign key to movie_id of Movie

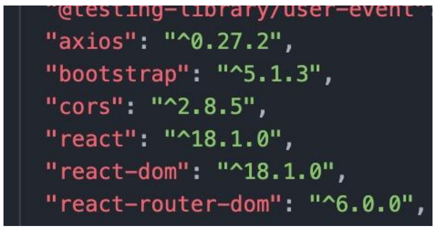
Movie_review(review_id, movie_id, customer_name, rating, review_message, spoiler, net_like_count, review_date)
customer_name foreign key to customer_name of Customer
movie_id foreign key to movie_id of Movie

Deleted_movie(deletion_id, employee_name, movie_title, production_year, director, avg_rating, genre, price, deletion_date)

6. Project Implementation Details

Backend: We wrote the backend with Java and used Spring framework. For the database system we used MySQL and connected it to the Java code using JDBC. With the help of Spring and Lombok libraries, we created SQL statements within the Java code. To connect the application to the database, we hardcoded the schema name, port no, MySQL username and password. After creating the connection between the database and the application, Spring library allowed us to use many database operations from the backend code such as specifying constraints, automatically generating ids, etc. We used Entity-Repository-Service-Controller architecture in the backend. Using the repositories, we were able to modify the database. The repositories were created as extensions to the JDBC repository.

Frontend: We wrote it using react.js. We have used bootstrap version 5.1.3 to design our components which are created using HTML5 and CSS3. By using React, we automatically used libraries which are Babel, Webpack. Babel is used to convert the ES6 JS code into ES5 so that it can run even on old browsers and webpack is used to decrease the size of the generated single html page. For navigation, we have used React Router and React Router Dom v6. This allowed us to navigate through pages and also add them to history so that users can also revert the navigation. Finally for connecting with the backend, we have used Axios which is a 3rd party library to handle http requests efficiently and easily. The UI is designed in an event driven fashion with the help useReducer and useContext hook so that the ui can dispatch events and the state can update itself accordingly. The dependancies we used are given below:



```
"@testing-library/user-event"
"axios": "^0.27.2",
"bootstrap": "^5.1.3",
"cors": "^2.8.5",
"react": "^18.1.0",
"react-dom": "^18.1.0",
"react-router-dom": "^6.0.0",
```

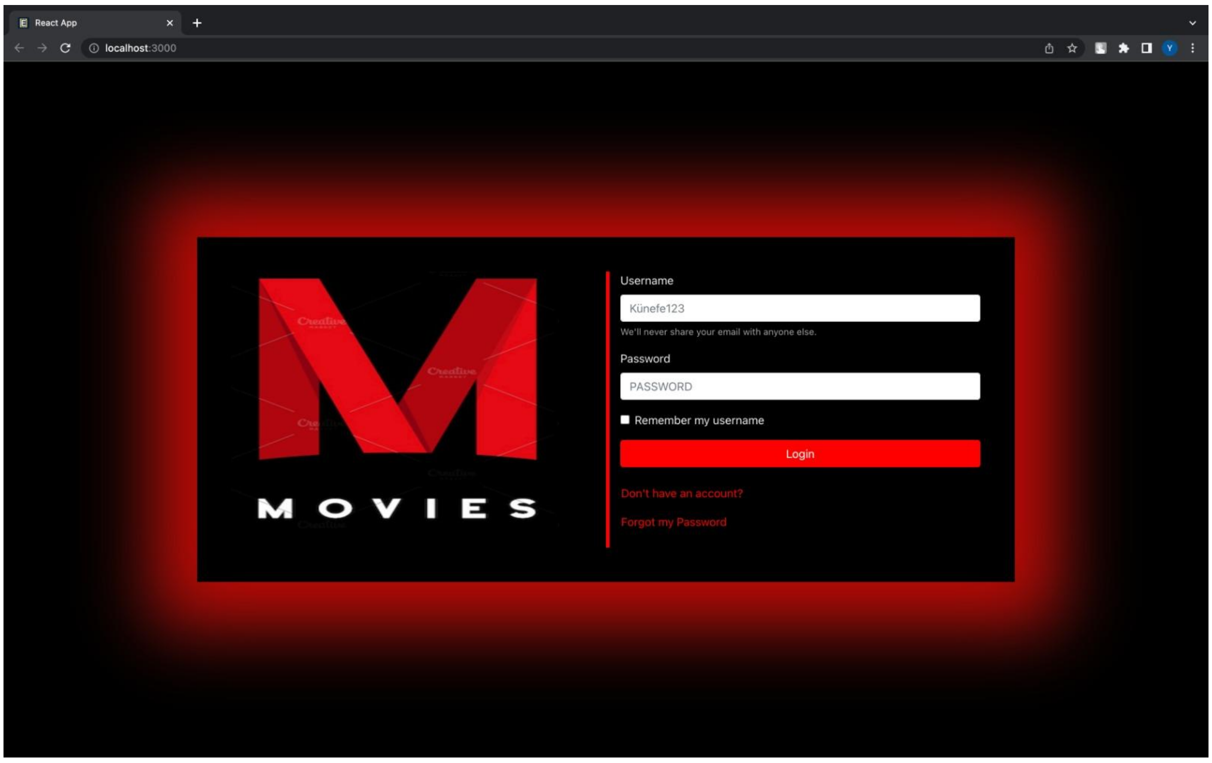
7. Advanced Database Components

Constraints: We used many constraints while forming our tables. For instance, movie_id in movie, card_id in card, actor_id in actor, request_id in movie_request, deletion_id in delete_movie and review_id in movie_review are automatically incremented using AUTO_INCREMENT as they're valid as long as they're unique and by auto-incrementing them we always get unique ids. In most of our entities, we used NOT NULL constraints for necessary attributes. We used the CHECK constraint in the movie entity to make sure that the price of the movie is not negative. In movie_review, we used CHECK constraint to make sure rating attribute is not negative and less than or equal to 10 which is the threshold for the rating. Also, in movie_review, we used the CHECK constraint to make sure that the spoiler rating is either 0 (not a spoiler) or 1 (is a spoiler).

Trigger: We used a trigger construct to automatically update the rating of a movie after a new review is given to that movie. It checks the rating of each review and calculates the average rating as the review of the movie.

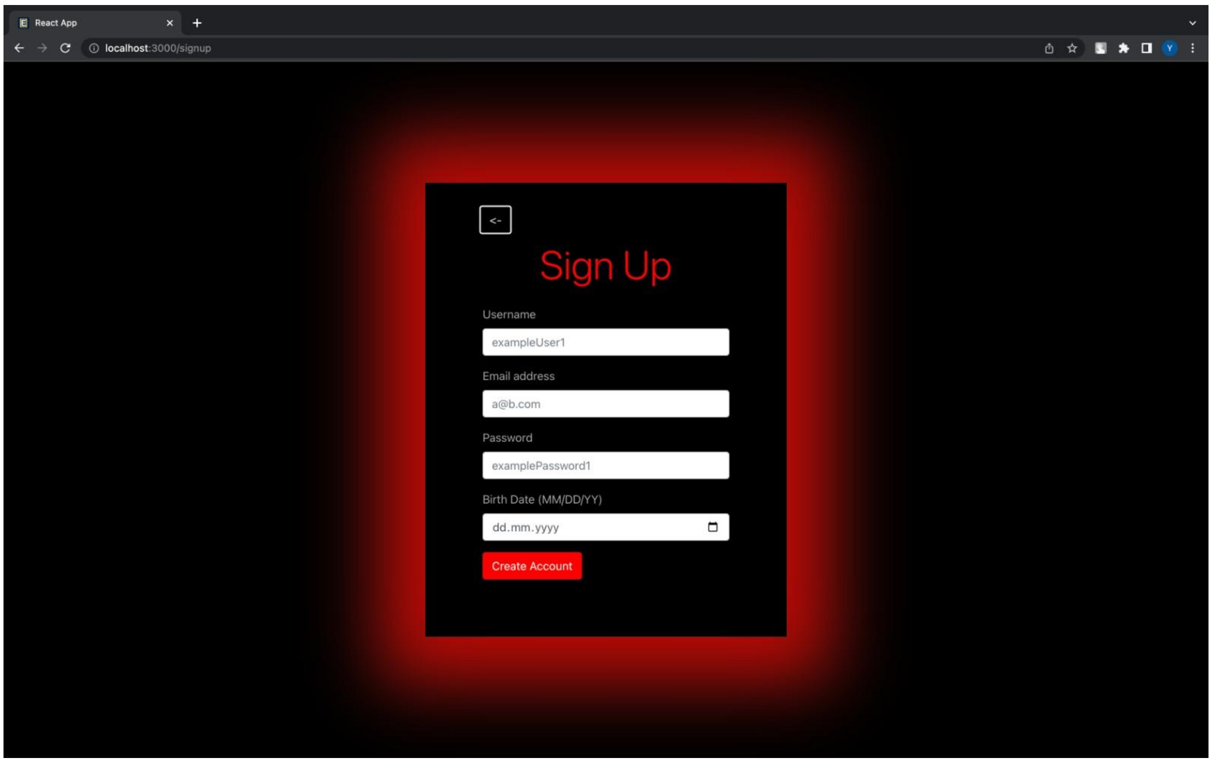
8. User's Manual

Login Page:



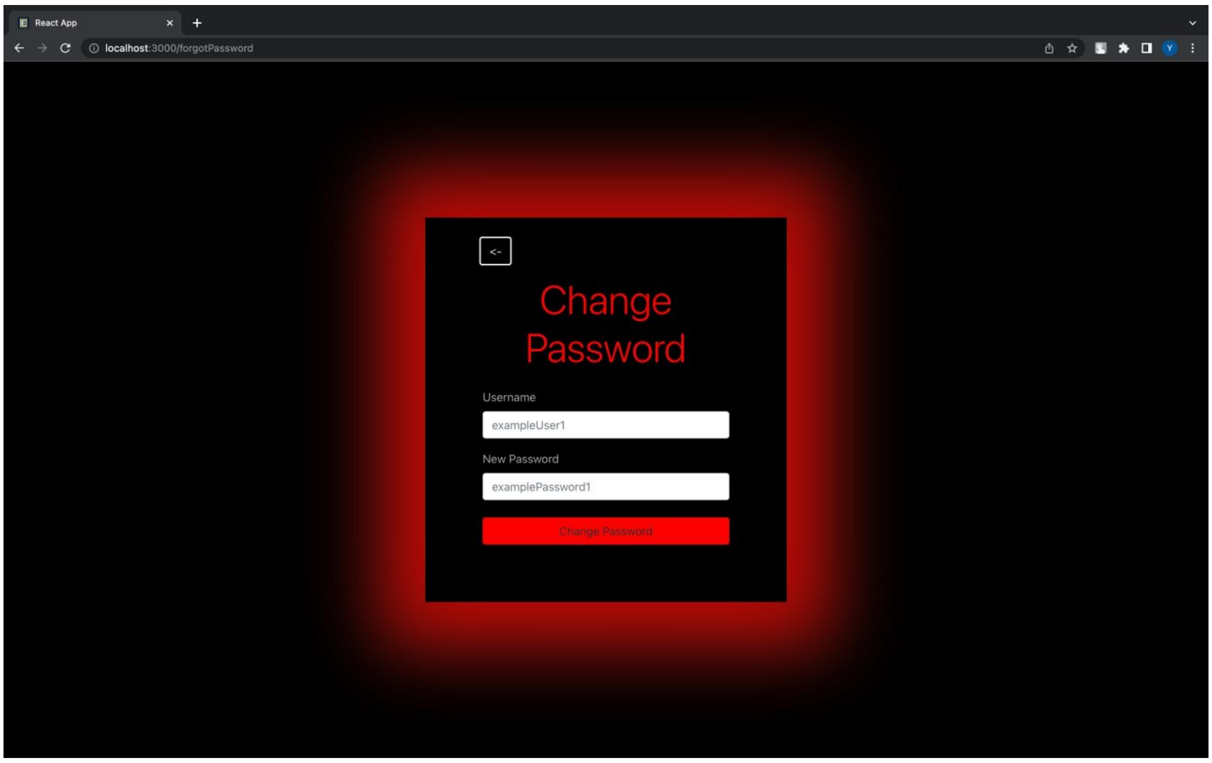
In the login page if a user has an existing account, he/she enters the username and password and clicks the red "Login" button to login to the system. If any of the text fields are empty or the inputs are not existent in the database, the user is alerted with a pop-up saying unsuccessful login. Else, "Successful Login" pop-up is shown and the user enters his/her account. There are 2 account types in the system, employee or customer. System automatically figures out the account type during login. New users can click the “Don’t have an account?” button and are directed to the signup page. If the user forgets his/her password, he/she can click the “Forget my password” button to be directed to the change password page where he/she can change his/her password.

Signup Page:


React App
localhost:3000/signup
<-
Sign Up
Username: exampleUser1
Email address: a@b.com
Password: examplePassword1
Birth Date (MM/DD/YY): dd.mm.yyyy
Create Account

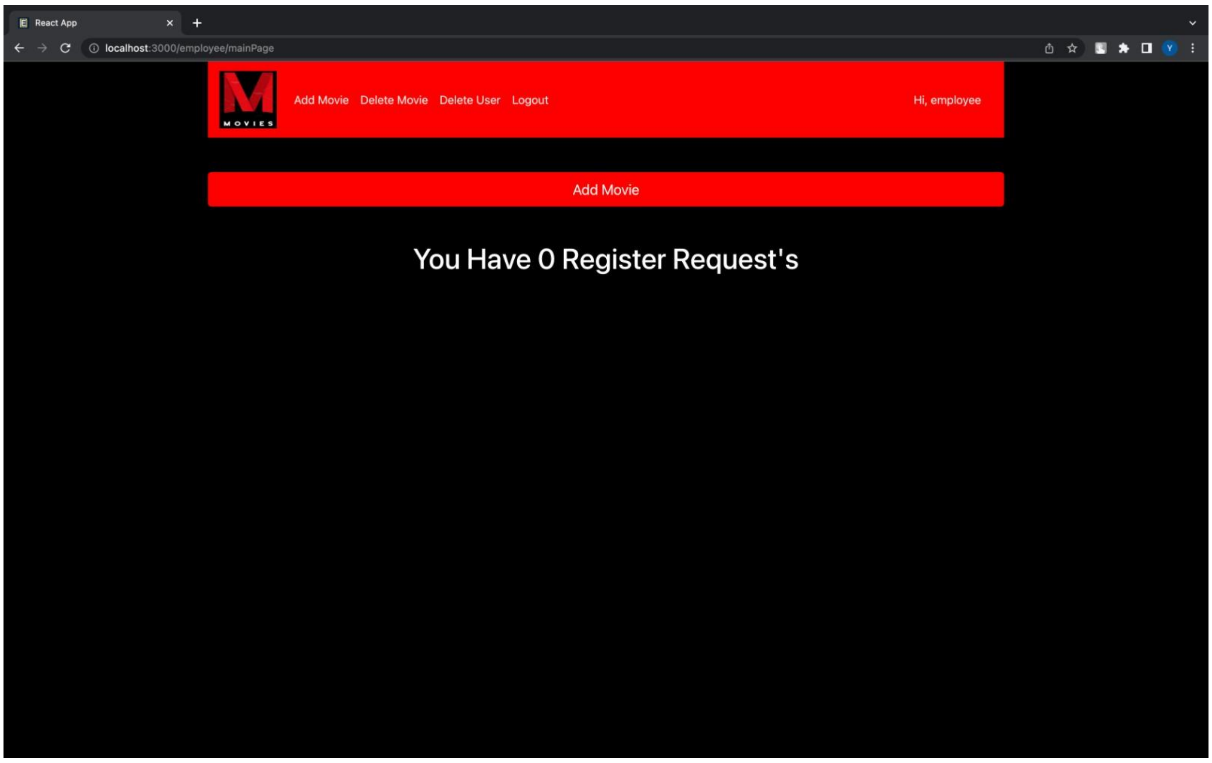
To create an account, the user fills the given text fields in the appropriate format and clicks the create account button. If any text fields are empty or the input is not in the correct format, the system informs the user by displaying a pop-up. After creating an account successfully, the user is directed back to the login page and shown a pop-up saying successful pop-up.

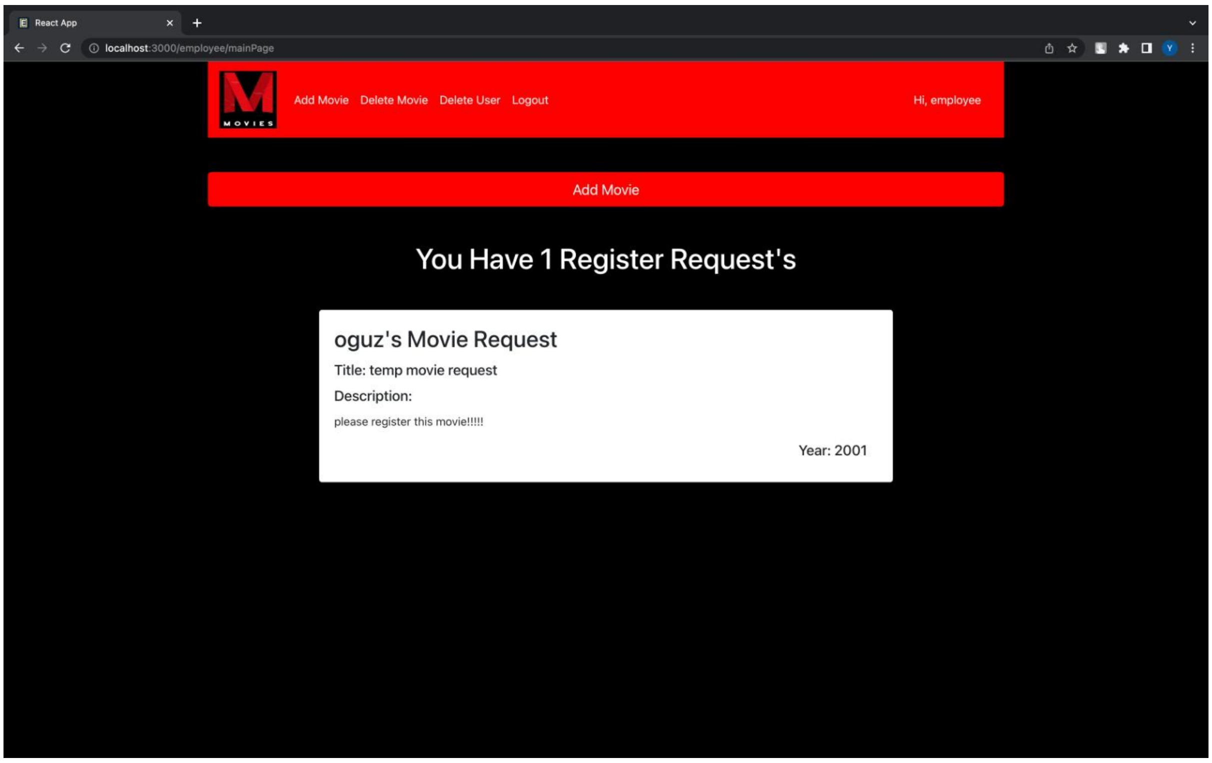
Change Password Page:



In this page, the user can change his/her password by filling the given text fields and clicking the change password button. In case that any text fields are left empty or if any accounts cannot be found, the user is informed by a pop-up. If the operation is successful, the user is directed back to the login page.

Employee Add Movie Page:




localhost:3000/employee/mainPage

Add Movie Delete Movie Delete User Logout Hi, employee

Add Movie

You Have 1 Register Request's

oguz's Movie Request

Title: temp movie request

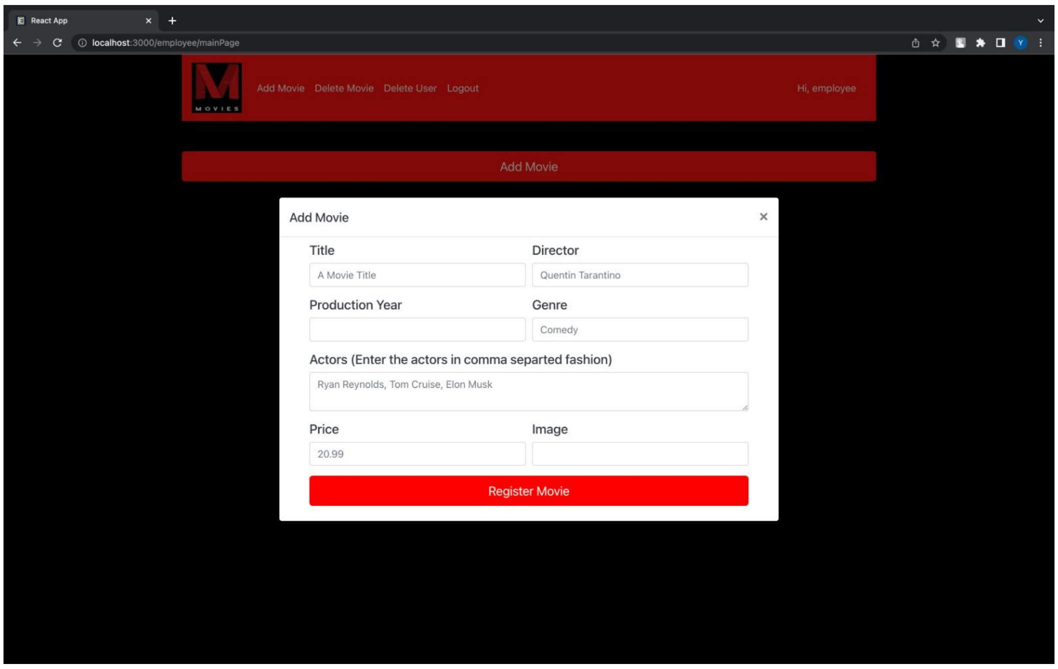
Description:

please register this movie!!!!!

Year: 2001

This page is only available for employee-type accounts. After logging in, the user is directed to this screen. Using the navbar, the user can switch between pages. In this page, the employee sees the movie requests made by customers. To add movies, the user clicks the add movie button and the and movie modal is displayed. Employees can add movies with or without any movie requests. First image shows the add movie page without any movie requests and the other image shows the add movie page with listed movie requests.

Add Movie Modal:


localhost:3000/employee/mainPage

Add Movie

Title: A Movie Title

Director: Quentin Tarantino

Production Year:

Genre: Comedy

Actors (Enter the actors in comma separted fashion): Ryan Reynolds, Tom Cruise, Elon Musk

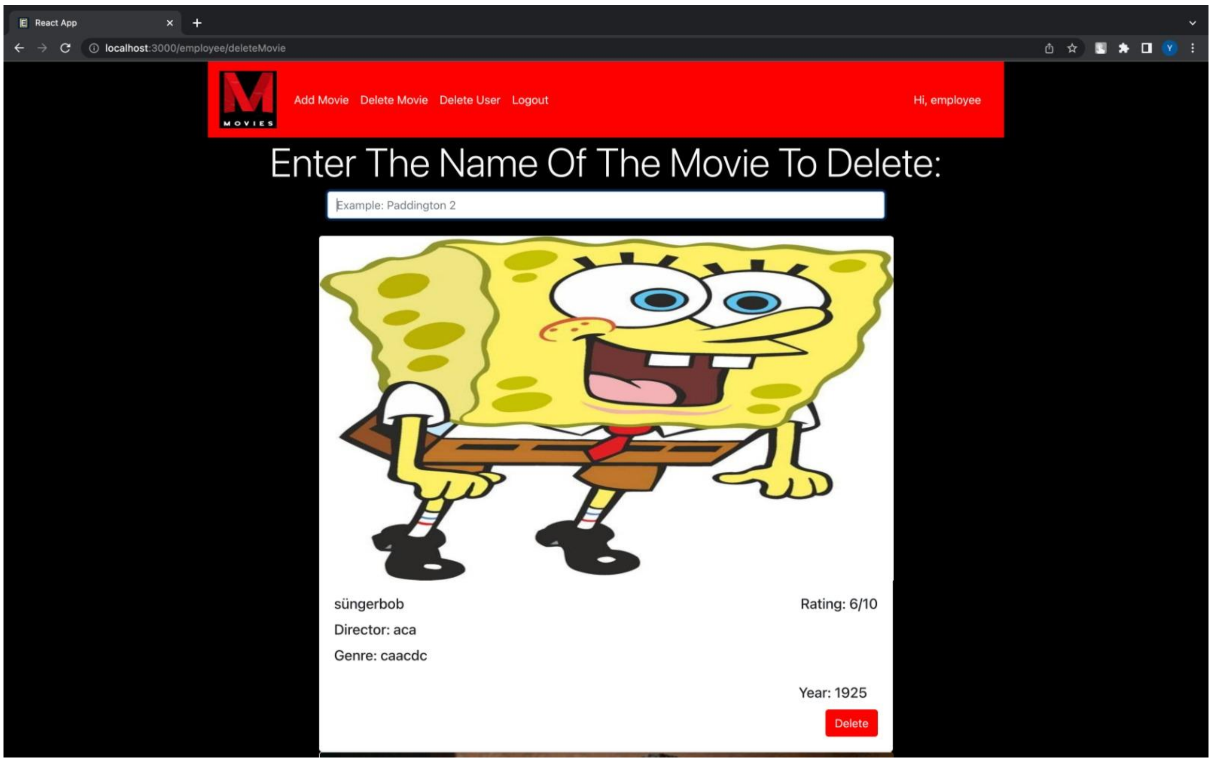
Price: 20.99

Image:

Register Movie

After clicking the add movie button, the employee-type user is displayed this modal. According to the requests or by choice, the user can add movies. To add the movie, the user fills the form with proper inputs. The price must be a positive number and the image must be an url. If any text fields are empty or inputs are in the wrong format, the user is alerted with a pop-up informing failed operation. If the operation is successful, the user is directed back to the add movie page. If the employee-type user adds a movie that is requested by a customer which is listed in the add movie page, the request disappears as it is added.

Employee Delete Movie Page:



Enter The Name Of The Movie To Delete:

Example: Paddington 2

süngerbob

Rating: 6/10

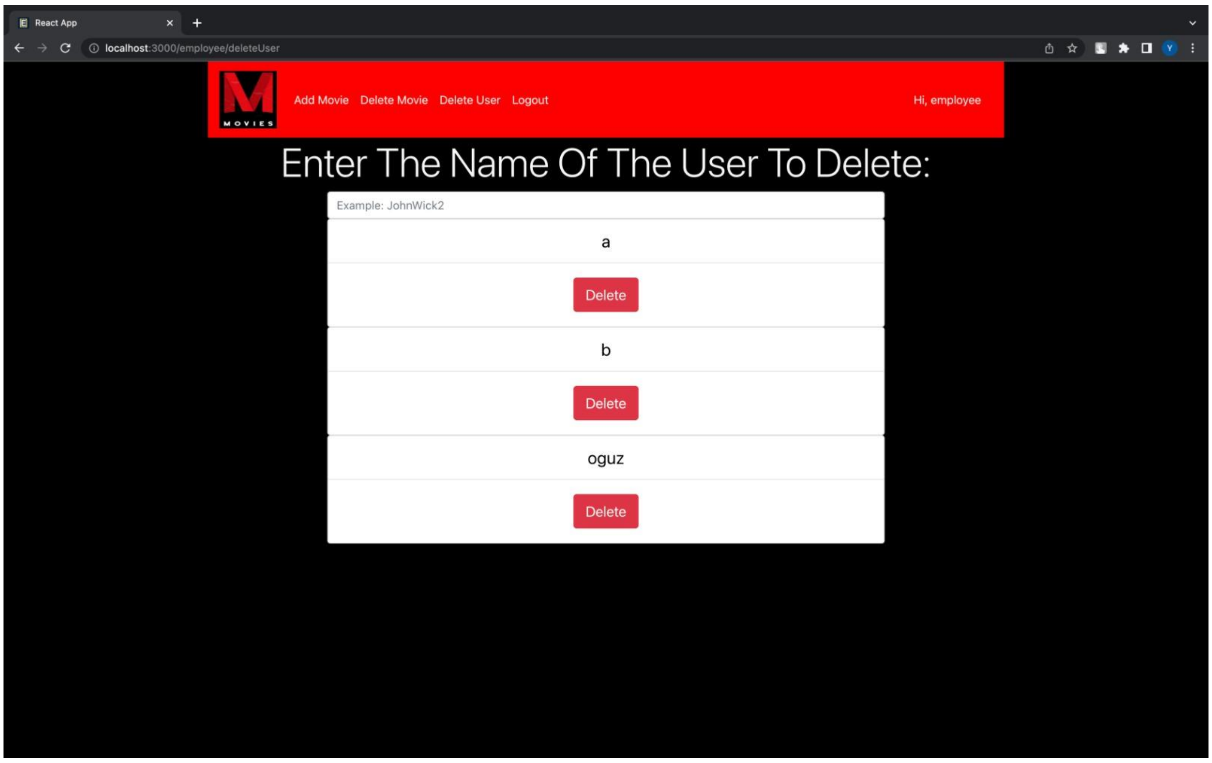
Director: aca

Genre: caacdc

Year: 1925

The delete movie page is only available to employee-type users. In the page, all movies that are in the system are listed with a delete button in the below-right corner. The user can search a certain movie by typing its movie title on the search bar. If a user opts to delete a movie by clicking the delete button, the movie is deleted from the system and disappears from the page.

Employee Delete User Page:



Enter The Name Of The User To Delete:

Example: JohnWick2

a

Delete

b

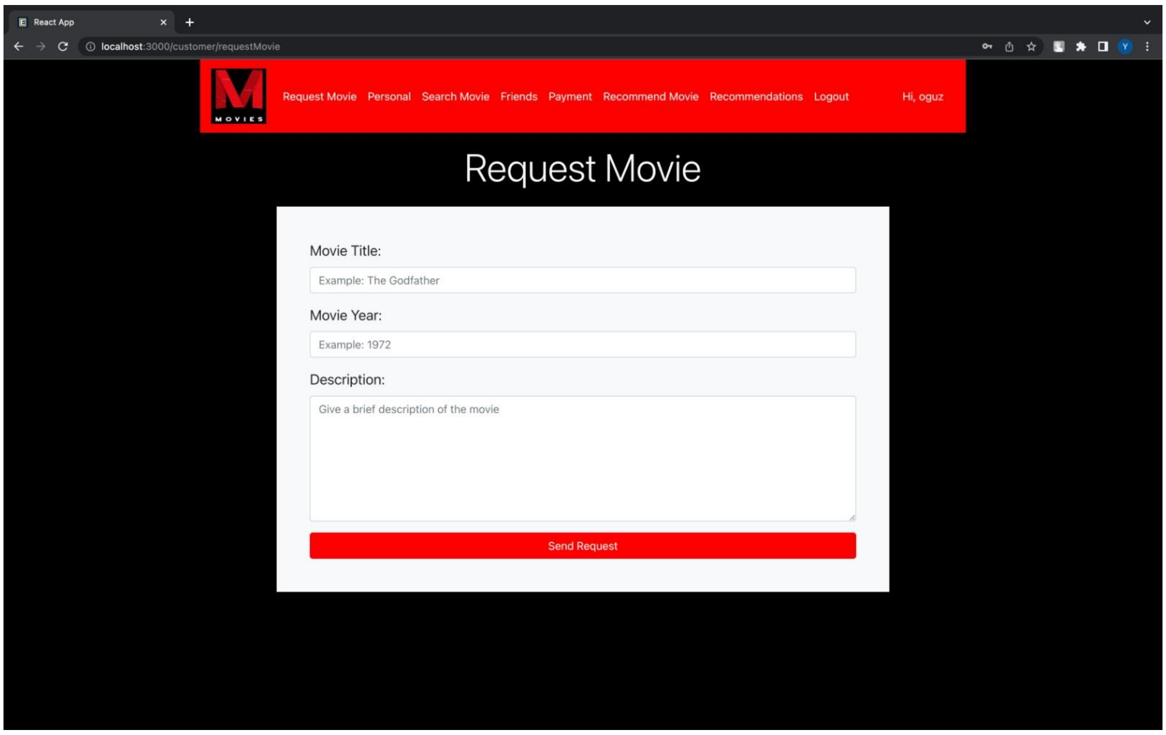
Delete

oguz

Delete

Similar to the delete movie page, the delete user page is only available to employee-type accounts. On the page, all users in the system are displayed. The user can use the search to search users by their username and if opts to delete the user, clicks the delete button. Then the user is deleted from the system and disappears from the screen.

Customer Request Movie Page:



Request Movie

Movie Title:

Example: The Godfather

Movie Year:

Example: 1972

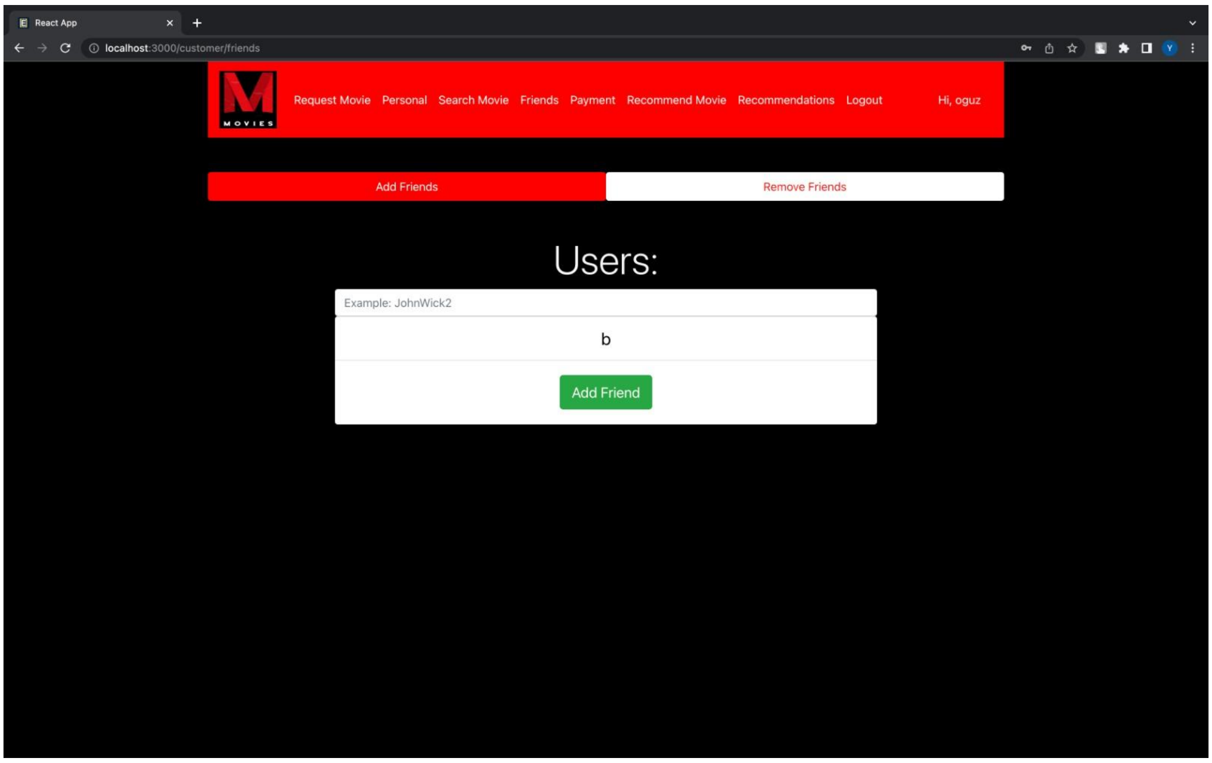
Description:

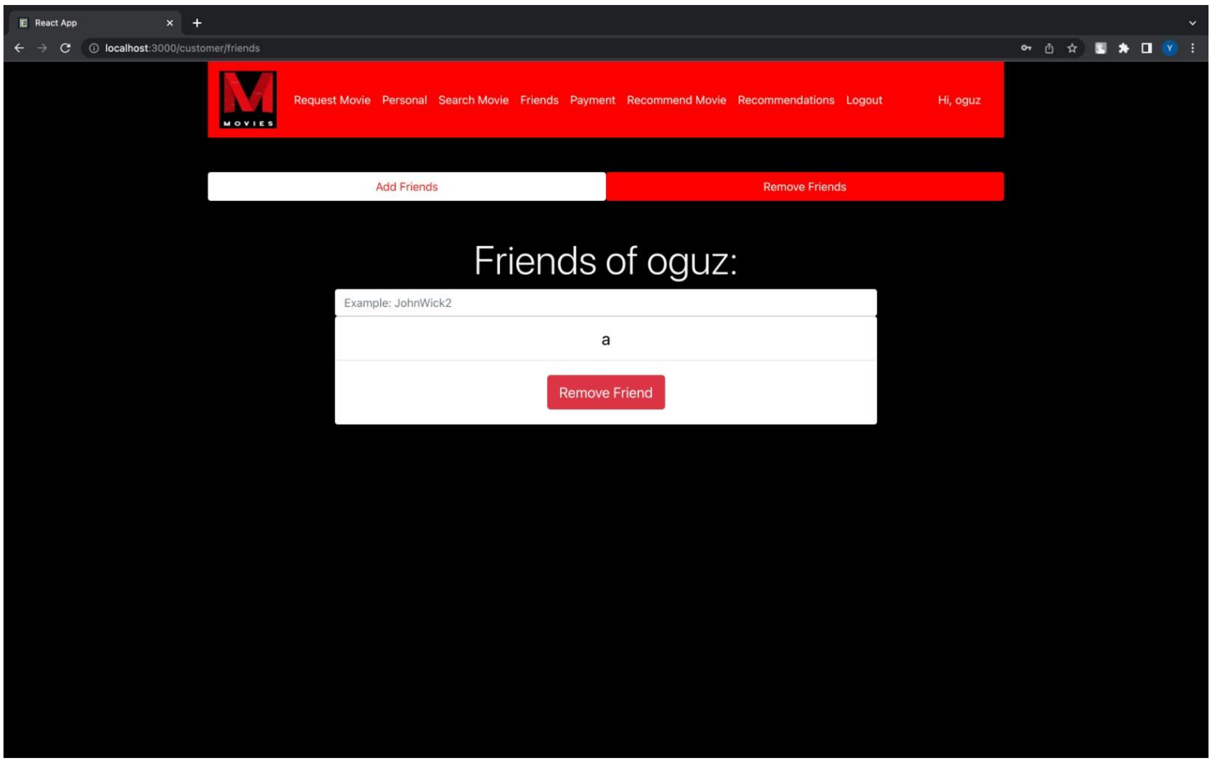
Give a brief description of the movie

Send Request

The request movie page is displayed in customer-type accounts. If the user wants to request a movie, he/can fill the form and click the send request button. If all text boxes in the form are filled in the correct format, the request is sent to the system to be displayed to employees, else the user is alerted by a pop-up informing the failed operation.

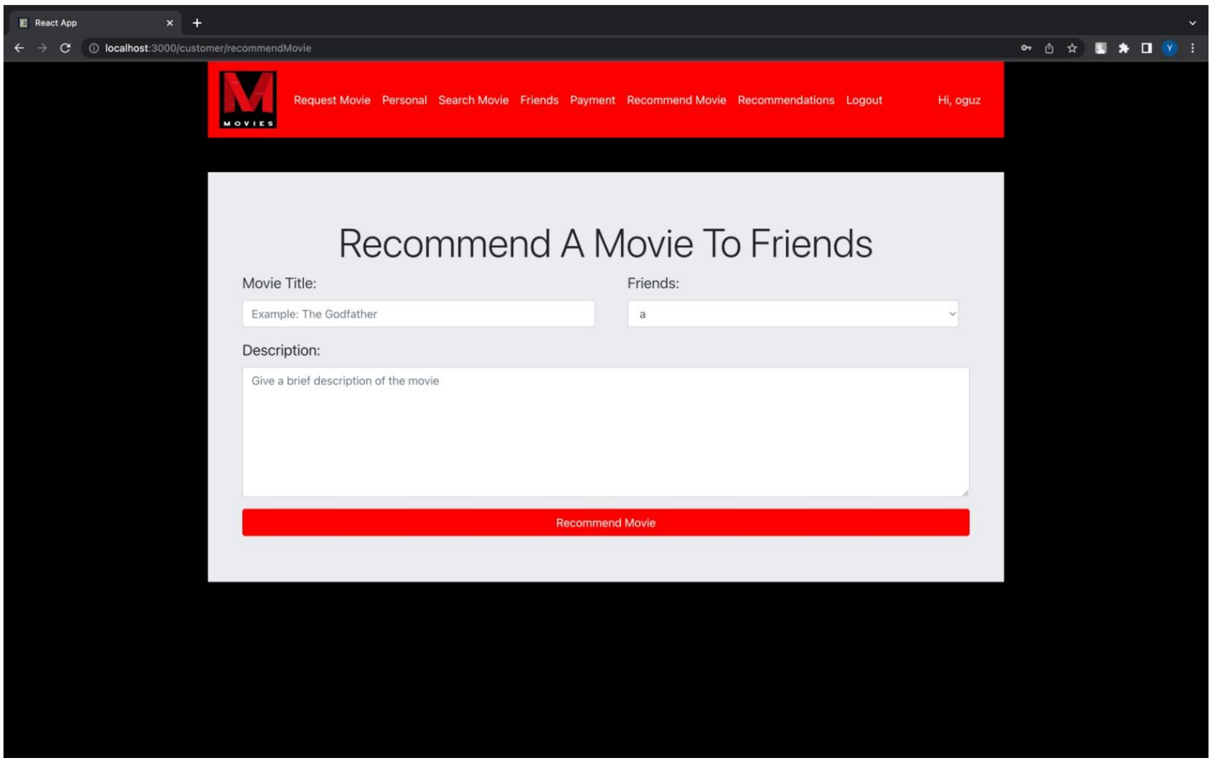
Customer Friends Page:





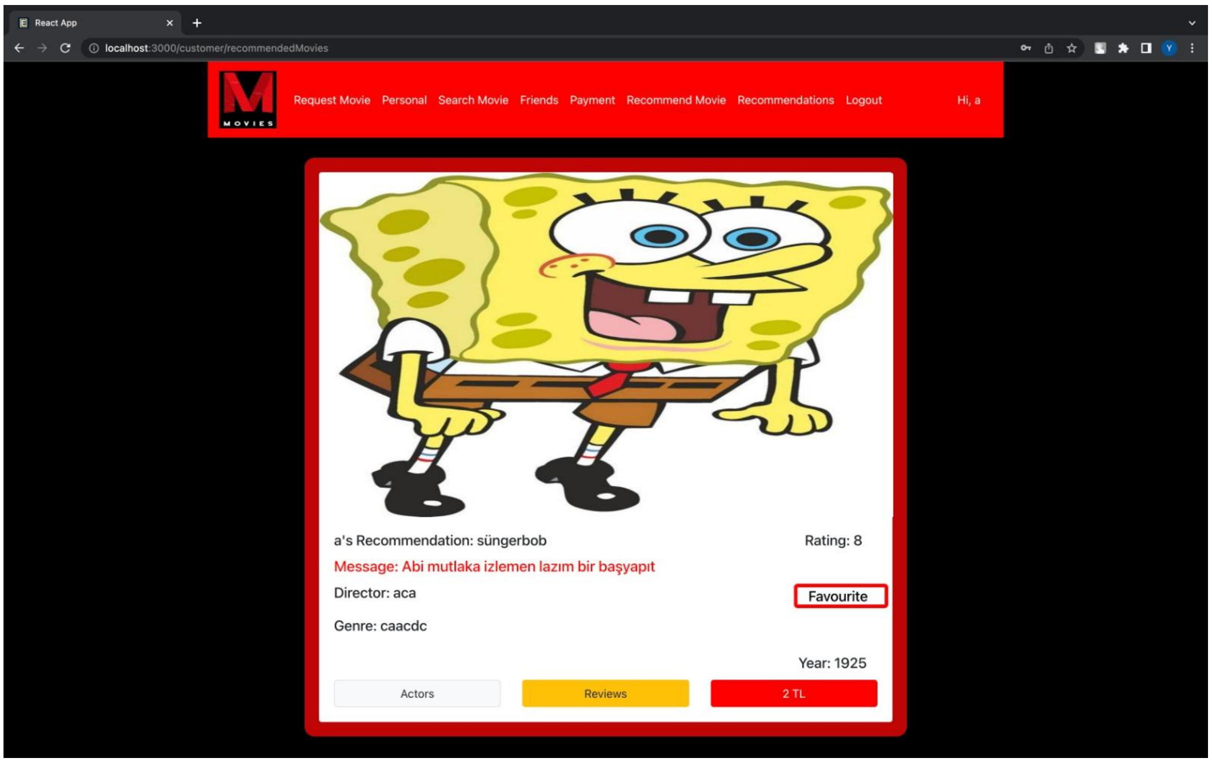
The friends page is only available to customer-type accounts. The user can switch between add friends and remove friends screen using the buttons below the navbar. In the add friends page all users are displayed to the user. Users can filter the user by typing the username of desired user in the search bar. If the user opts to add a user as his/her friend, he/she clicks the add friend button below each listed user. Then, that user is removed from the list in add friend screen and added to the list in remove friends. The remove friends page displays the list of friends the user has. The user can again filter the list by entering the username of the desired user. By clicking the remove friend button under each user, the user can unfriend a user which removes the user from this list and places it back to the list in the add friends page.

Customer Recommend Movie Page:



The recommend movie page is only available to customer-type accounts. In this page, a user can recommend a movie to one of his/her friends by filling the form given. The user can select the user/friend from the list given when clicking the text field dedicated to friends or can enter the username of the friend. If the user has no friend, the user cannot fill the friends text box and hence the operation fails when the recommend movie button is clicked. The operation fails if any text fields are empty or the movie entered does not exist in the system and the user is informed by a pop-up displaying failed operation.

Customer Recommendations Page:



a's Recommendation: süngerbob

Rating: 8

Message: Abi mutlaka izlemen lazım bir başyapıt

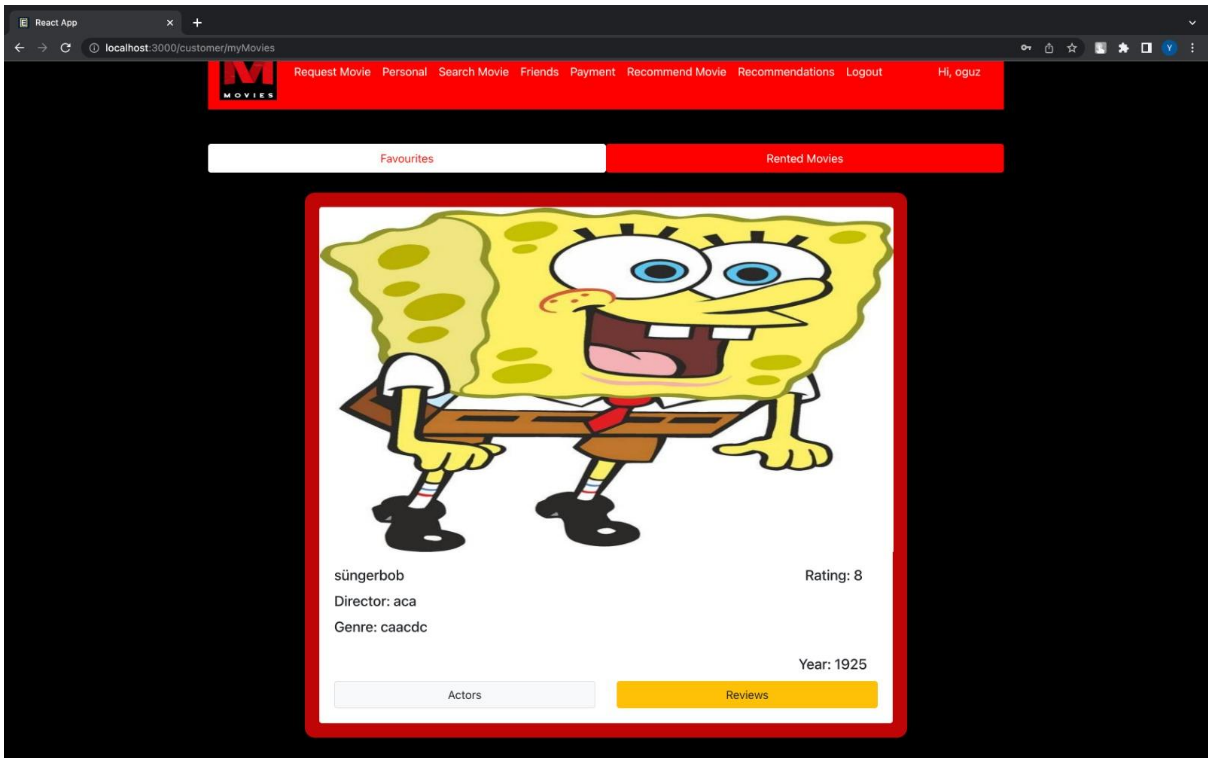
Director: aca

Genre: caacdc

Year: 1925

The recommendations page is only available to customer-type accounts. In this page, the user can display the recommendations made to him/her in the form of movie cards. If there are no recommendations, the page displays no recommendations. The user can interact with the operations on the movie card mounted on the screen by clicking the button on the card, the movie card is discussed later in the manual.

Customer Personal Page:



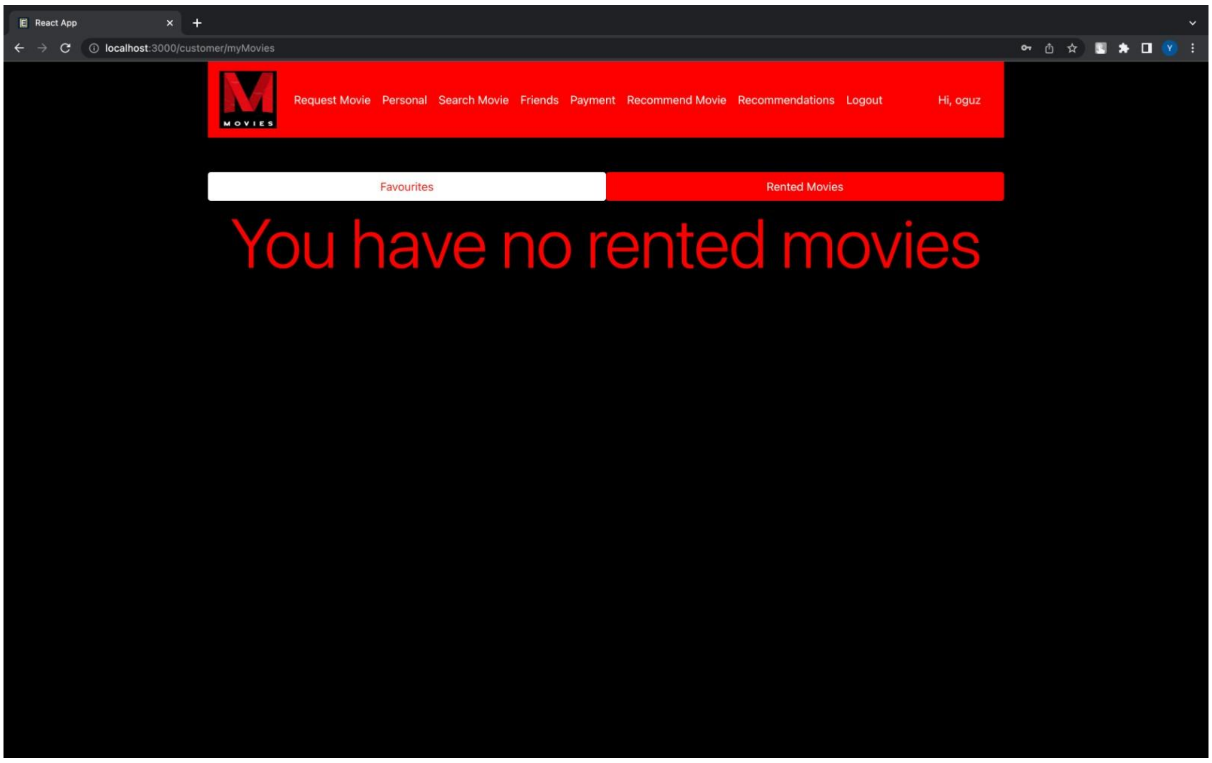
süngerbob

Rating: 8

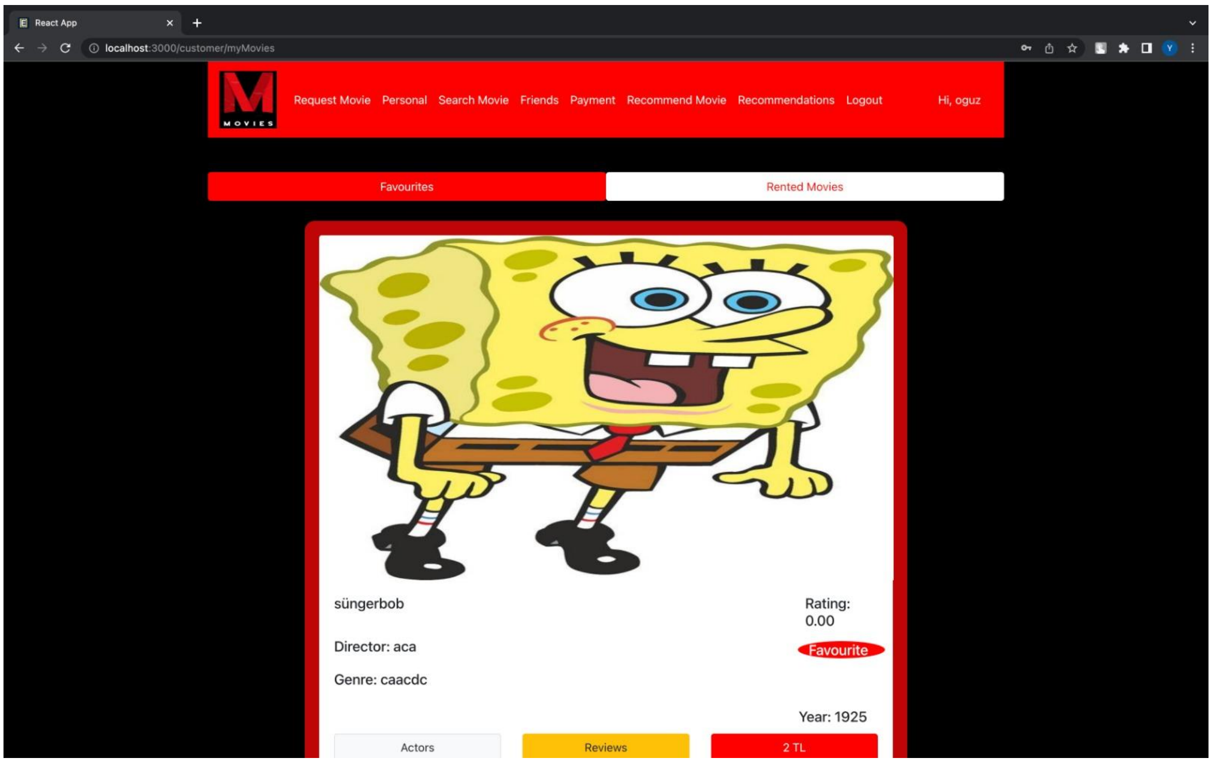
Director: aca

Genre: caacdc

Year: 1925

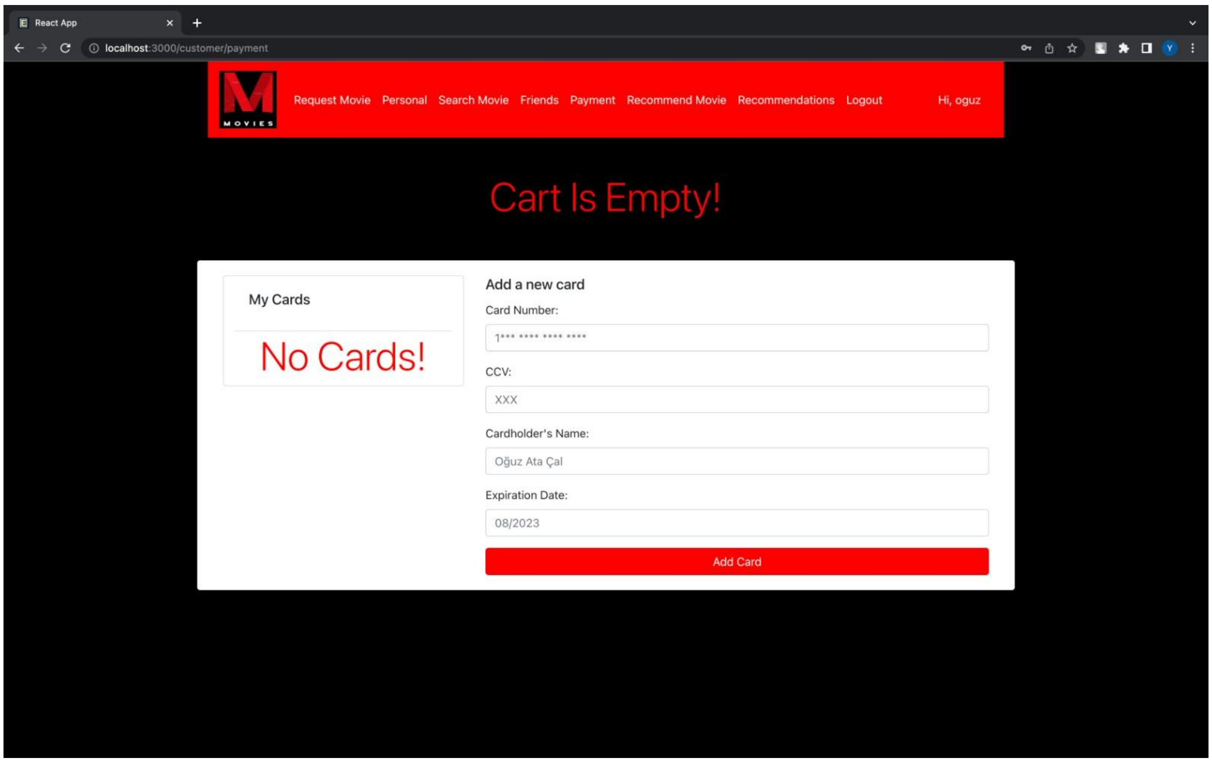


You have no rented movies



The personal page is only available to customer-type accounts. In this page, the user can display his favorite movies and the movies that he/she has rented. The user can switch between the favorites screen and the rented movies screen by using the buttons below the navbar. In the rented movies screen, movie cards of the rented movies are displayed. If no movies are rented, the screen displays “You have no rented movies”. In the rented movies screen, the rent button of the movie cards are removed as they’re already rented by the user. The rented movies that are listed are removed after 1 month, which is the deadline of the rent and automatically removed from the list. Other than that, the user can interact with the movie card, which is discussed later in the manual. In the favorites screen, all movies marked as favorite using the favorite button in the movie card are displayed. They can be removed from the screen by clicking the favorite button again. Other operations of the movie card can be interacted with by the user using the buttons on the card.

Customer Payment Page:



Cart Is Empty!

My Cards

No Cards!

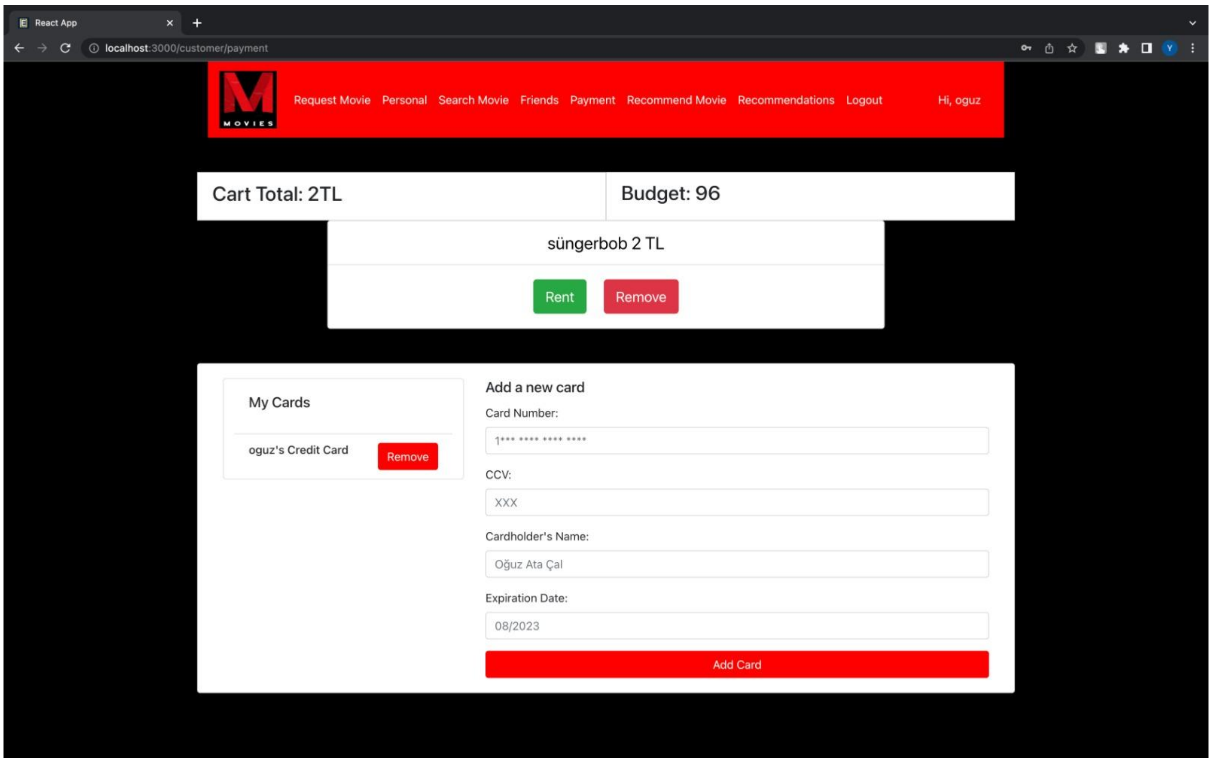
Add a new card

Card Number: 1*** **** **** ****

CCV: XXX

Cardholder's Name: Oğuz Ata Çal

Expiration Date: 08/2023



Cart Total: 2TL

Budget: 96

süngerbob 2 TL

My Cards

oguz's Credit Card

Add a new card

Card Number: 1*** **** **** ****

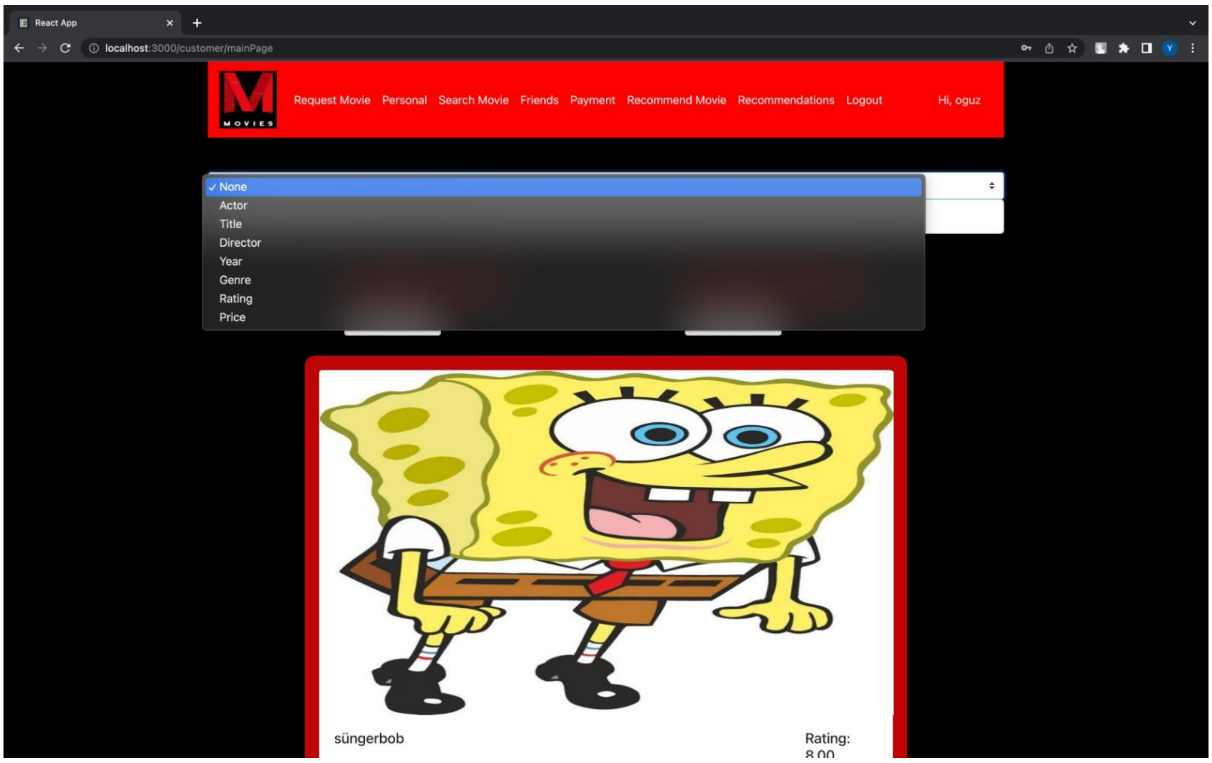
CCV: XXX

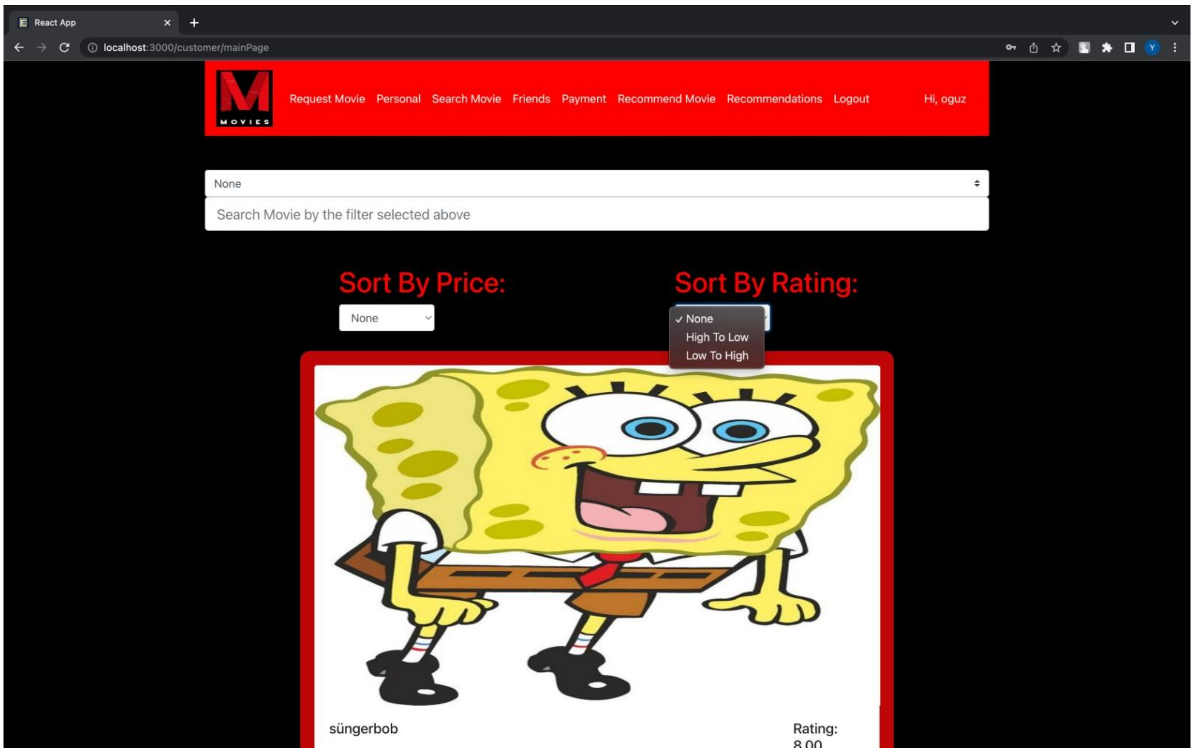
Cardholder's Name: Oğuz Ata Çal

Expiration Date: 08/2023

The payment page is only available to customer-type accounts. The movies added to the cart by the user are displayed in the page along with the card form. If the cart is empty, the screen displays “Cart Is Empty!”. The user can remove the movies listed in the cart by clicking the remove button below each movie. The cart total and balance of the user is displayed above the cart. If the user clicks the rent button under a listed movie, the price of the movie is subtracted from both the cart total and the balance of the user. Below the cart is the card form. On the left-hand side of the form are the cards added by the user. Users can remove a card by clicking the remove button below the card, and the card is removed from the system and the list. The user can add a card using the form on the right-hand side of the screen. The user should fill all text boxes in the correct form to add the cart, else the user is shown a pop-up informing failed operation after clicking the add card button.

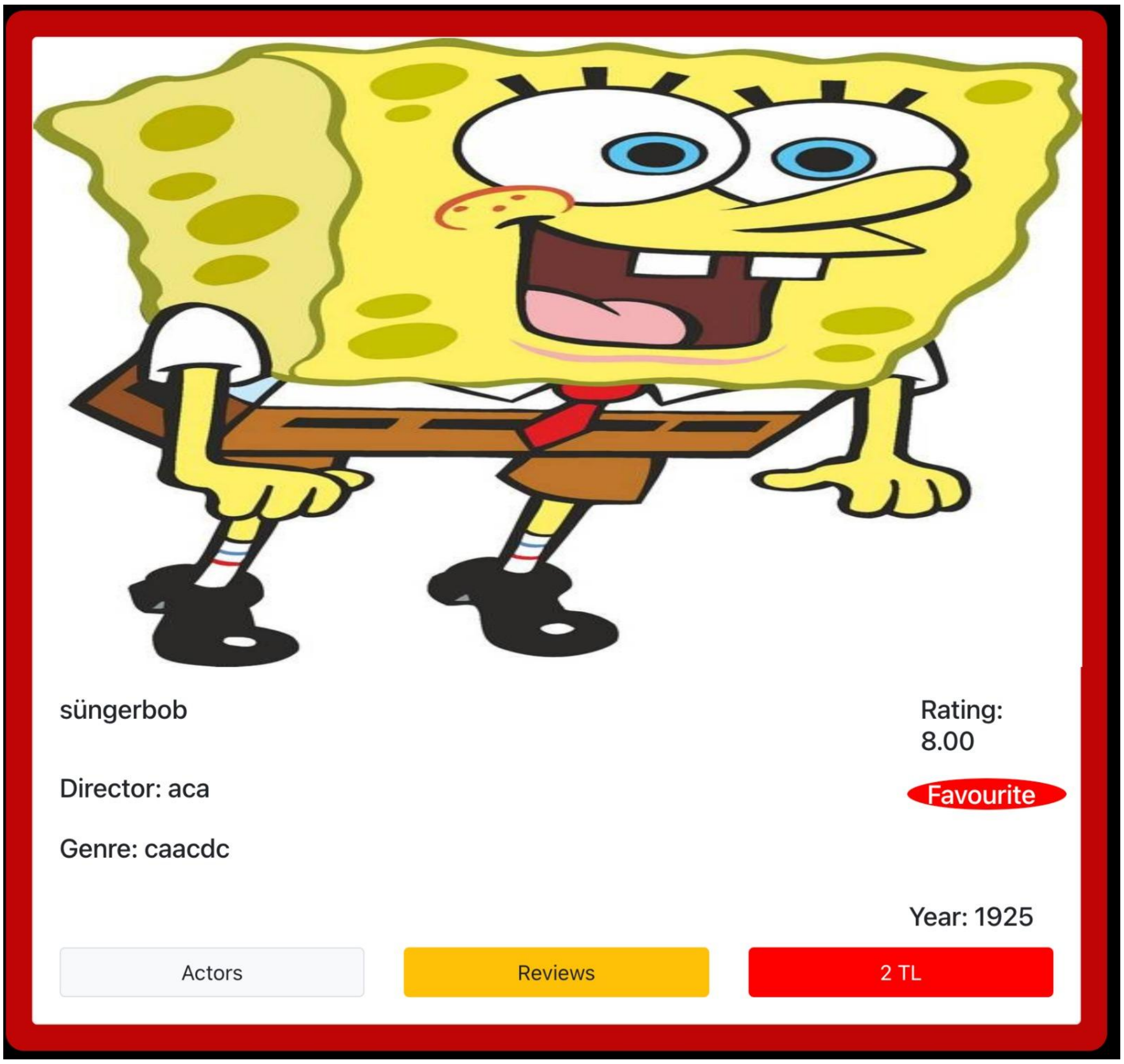
Customer Search Movie Page:





The search movie page is only available to customer-type accounts. In this page, the user can see all the movies in the system. Each movie is displayed in a movie card, which the user interacts to rent, review or see the actors. The movie card is discussed later in the manual. The user can use the search bar above, by entering a prompt and selecting the filter by clicking the select box. Also, users are given the option to list the movies by rating, and by price. The options are, high to low or low to high for both rating and price. If a movie is deleted or added by an employee, the list is updated automatically by the system. If the user selects a numerical filter from the select box, he/she can give a range in the form of (x-y) and the system will list the movies in the range of at least x and at most y.

Movie Card:


süngerbob

Director: aca

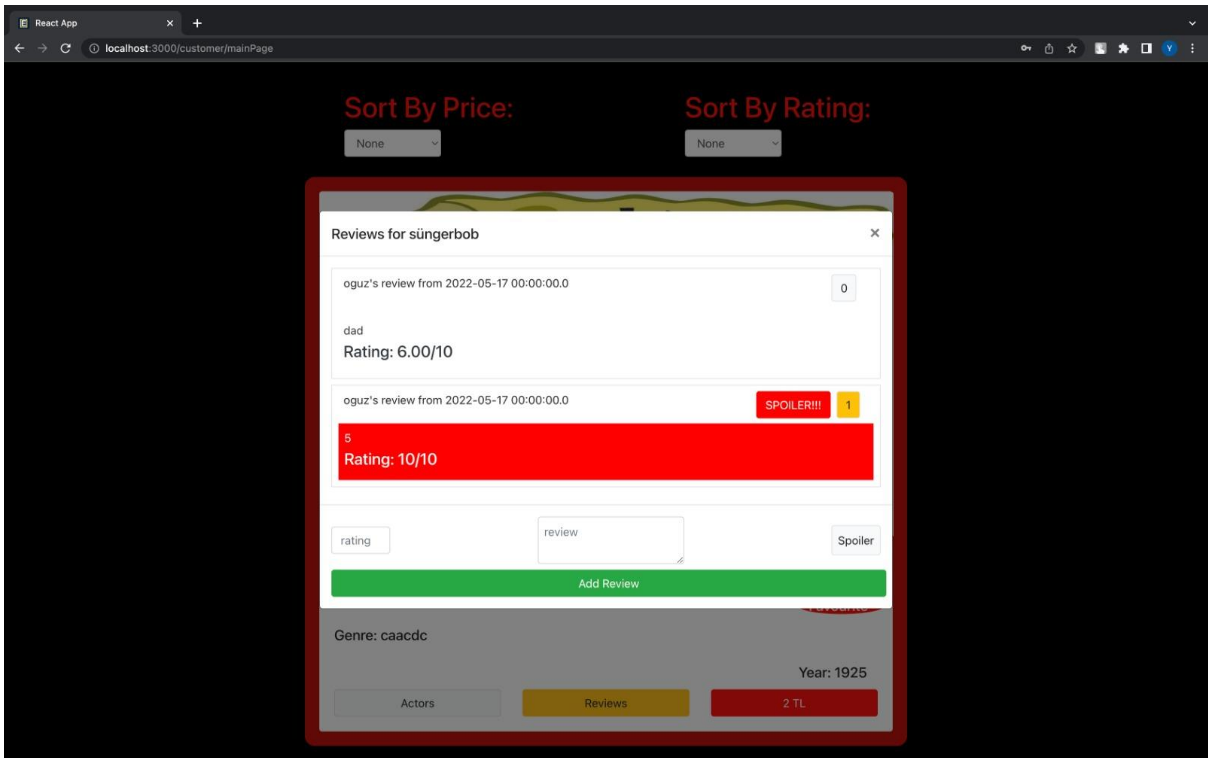
Genre: caacdc

Rating: 8.00

Year: 1925

The movie card is used in almost every page where movies are listed. The information (director, genre, movie rating, movie title, production year) about the movie is displayed on the card. The user can mark the movie as a favorite by clicking the favorite button on the right which adds the movie to the list in the favorite screen on the personal page of the user. Also, users can see the actors of the movie by clicking the actors button, which displays the actors in a pop-up. The user can see the rent price of the movie on the button on the right-hand side. By clicking that button, the user adds the movie to his/her cart which is displayed in the payments page. To write a review and/or see reviews made by other users, the user can click the reviews button which displays the movie reviews modal which is discussed next.

Movie Reviews Modal:


Reviews for süngerbob

oguz's review from 2022-05-17 00:00:00.0

dad

Rating: 6.00/10

oguz's review from 2022-05-17 00:00:00.0

5

Rating: 10/10

Genre: caacdc

Year: 1925

The movie review model is only available to the customer-type accounts and is displayed when the review button on the movie card is clicked. The modal shows the list of reviews on the upper side and the review form on the lower side. The user can see all the reviews made by the other users about the movie. Each review has a like button which also displays the like count. The like button's color changes from white to yellow when clicked, which shows the review has been liked. The like can be taken back by clicking the like button again which turns its color back to white. The reviews may include spoilers and hence spoiler-marked reviews are displayed in red with a spoiler button above. The review message can be displayed by clicking the spoiler button above the message. Also, the rating of each review is displayed. The review form allows users to write reviews for a specific movie. The user can rate the movie between 0 and 10 and mark the review as a spoiler. By clicking the add review button, the review form is cleared and the review is added to the list displayed above. Each rating given for a review impacts the rating displayed on the movie card as it displays the average of all ratings and the system automatically calculates when a review is added.

9. SQL Queries for Topic Specific Functionalities

Note: Assume that the “input_attributes” are the inputs that will be taken from the user for a search operation

1a:
** Search by title, director, production year, genre or actor **

SELECT * FROM movie WHERE movie_title = input_title;

SELECT * FROM movie WHERE director = input_director;

SELECT * FROM movie WHERE production_year = input_year;

SELECT * FROM movie WHERE production_year = 2022;

SELECT * FROM movie WHERE genre = input_genre;

SELECT * FROM movie NATURAL JOIN acts NATURAL JOIN actor WHERE actor = input_actor;

1b:
** Filter the results by price or rating, and sort them **

SELECT * FROM movie ORDER BY production_year DESC;

SELECT * FROM movie ORDER BY avg_rating DESC;

SELECT * FROM movie ORDER BY price ASC;

SELECT * FROM movie WHERE avg_rating BETWEEN input_lower_bound AND input_upper_bound ;

SELECT * FROM movie WHERE avg_rating BETWEEN 8 AND 9;

SELECT * FROM movie WHERE price BETWEEN 2 AND 4;

1c:
SELECT * FROM movie WHERE movie_id = selected_movie_id;

1d:

Return date of the rented movie is automatically set to 1 month later.

1e:

Note: this_customer_name is the username of the customer that is willing to rent the movie, selected_movie_id is the selected movie's id and today is the date that the renting takes place.

INSERT INTO Rent VALUES (this_customer_name, selected_movie_id, today);

UPDATE Customer SET balance = balance - movie_price
WHERE customer_name = this_customer_name;

2. Movie Request

a. Search if the movie exists in the database

SELECT * FROM Movie m WHERE m.movie_title = title

b. Fill out the form with the requested movie's information

no associated query

c. Submit the form

INSERT INTO movie_request (request_id, movie_title, production_year, customer_name) VALUES (0, "Gora", 2004, "Example customer")

d. Employee receives the request form

SELECT * FROM movie_request

3. Movie Rating and Review

a. Select the movie

i. If it is a previously rented movie, select it from user's list

no associated query

ii. If it is not a rented movie, use search engine of the database

Ex: Search by genre: "SELECT * FROM movie WHERE genre = input_genre"

Search by title: SELECT * FROM Movie m WHERE m.movie_title LIKE %title%

Search by price range: SELECT * FROM movie WHERE price BETWEEN min AND max

b. Specify the rating out of a given threshold

Raiting thresholds is checked in the movie review form before submitting it

c. Write a review

no associated query

d. Publish the review on movie

INSERT INTO movie_review (review_id, net_like, raiting, review_date, review_message, spoiler, customer_name, movie_id) VALUES (0, 0, 10, 2022-05-17, “The film was excellent”, false, “Example customer name”, 2)

e. Update the movie’s overall rating

Assuming the movie_id of the movie is 10

UPDATE movie m

SET avg_rating = (SELECT AVG(rating) FROM movie_review WHERE movie_id = 10)

WHERE m.movie_id = 10