Final Project Report

MyMovieList

San Jose State University
CS 157A
Dr. Ching-seh Wu

Jiawei Zhang
Udaypal Singh
Adan Hernandez

Project Requirement

Project Description

Our proposed project is MyMovieList, a social media web application for both avid and casual moviegoers!

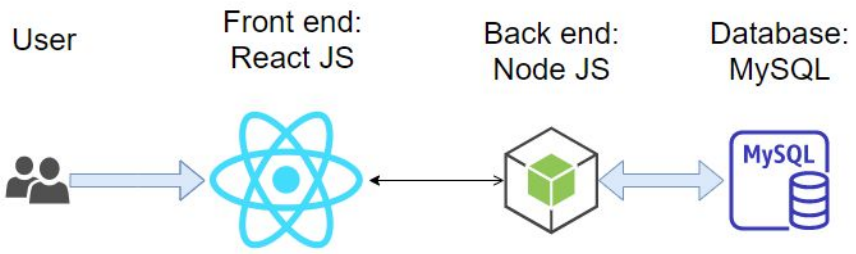
The purpose of MyMovieList is to give movie watchers a platform to show off all the movies and tv shows they have watched, as well as leave reviews for these movies and tv shows. Ultimately we want this application to be one that people will use for their movie inquiries and connecting people with similar interests.

At the moment, there is no technology out there for people to connect with other moviegoers or for people to keep track of what they have watched.Many tend to forget which movies they have watched, so using this web application will allow them to track the list of movies they have watched with the rating given by the user. It is almost positive that there has been a time where someone or yourself is trying to remember the name of a movie that they watched in the past. It's on the tip of their tongue, but they just can't remember. This then leads to an awkward pause or a disconnect between the people in the conversation. Our tool aims to solve this problem and help people connect in a much easier and comfortable way with each other. Often, people have the same interest as their friends, so people can look at their friends' movies list and look at the reliable rating and suggestions made by their friend. This application will be designed specifically for ease of use and appealing to the users.

The reason MyMovieList is the to-go application for your movie needs is because we put our customer first. We want it to be more than just a list of movies. We want it to be fun, interactive, and useful. We want people to connect with others around the world over the shared interests we know they all have.

The stakeholders for this web application will be mostly teens and young adults, along with movie enthusiastic people. Stakeholders can be anyone from a general population to a professional critic. The targeted audience for this web application is passionate movie and film viewers.

System Environment



Hardware and Software Used:

- HTTP Apache Tomcat
- Windows

RDBMS Used:

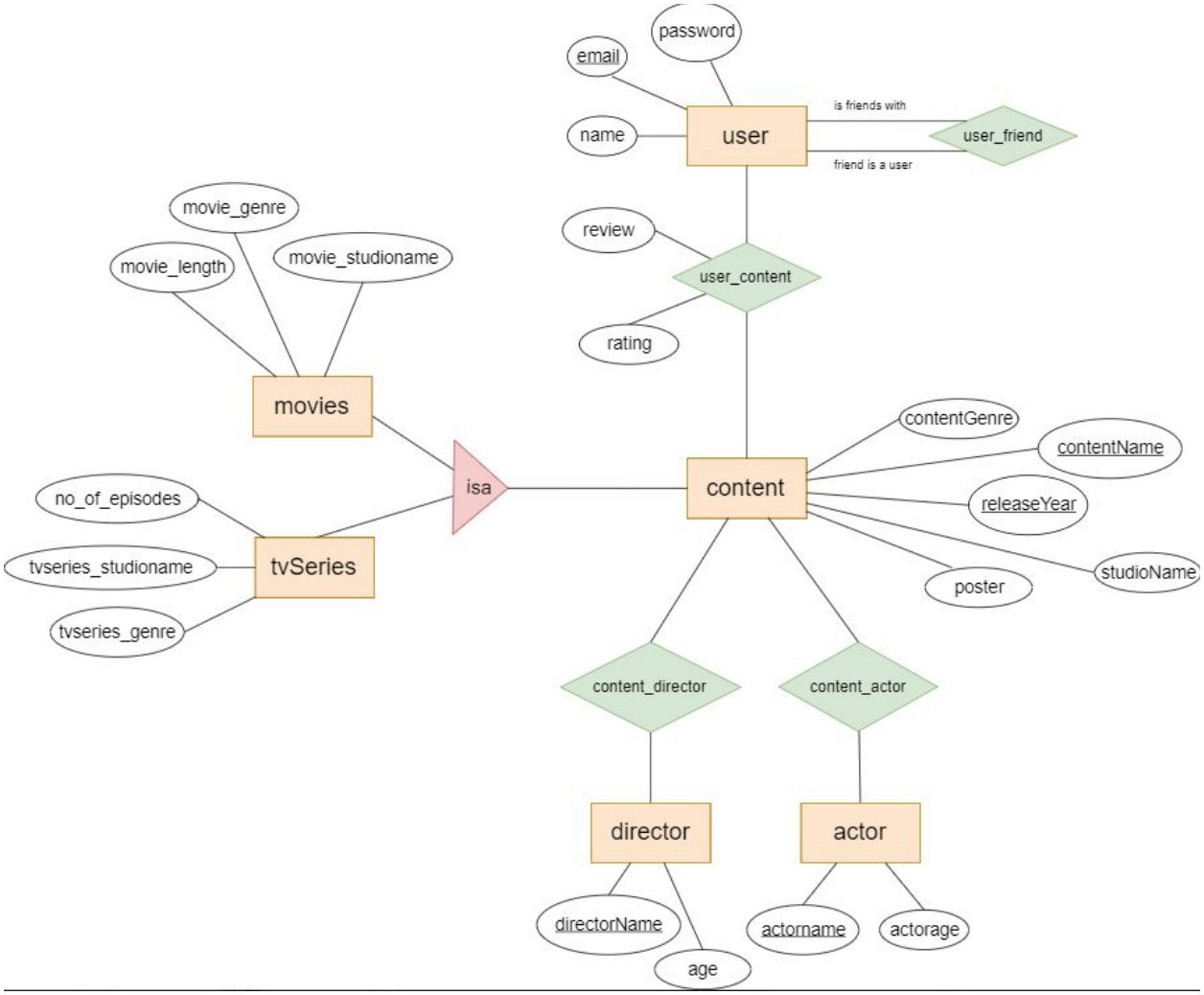
- MySQL

Application Languages Use:

- Javascript
- React JS
- CSS
- HTML
- Node JS

Project Design

MyMovieList ERD



Entity and attributes

- Entity: User
  - User(email, password, name)
  - Constraint: Email is primary key
  - User has access to website.
- Entity: Content
  - Content(releaseYear, contentName, contentGenre, studioName, poster)
  - Constraint: releaserYear and contentName are primary key.
  - Content has all the movie and TV Series information.
- Entity: Director
  - Director(directorName, age)
  - Constraint: directorName is primary key.
  - Director stores the information of directors.
- Entity: Actor
  - Actor(actorName, actorage)
  - Constraint: actorName is primary key.
  - Actor stores the information of Actor.
- Entity: Movies

    - Movies(movieName, movie_genre, movie_length, movie_studioname)
    - Constraint: movieName is primary key.
    - Movie stores the information of Movies.
- Entity: TvSeries
    - TvSeries(TvName, no_of_episodes, tvseries_studioname, tvseries_genre)
    - Constraint: TvName is primary key.
    - TvSeries store all the information TvSeries.
- Relationship: user_content
    - user_content(review, rating,email, contentName, releaseYear)
- Relationship: content_director
    - content_director(contentName, releaseYear, directorName)
- Relationship: content_actor
    - content_director(contentName, releaseYear, actorName)
- Relationship: user_friend
- Non-Trivial Functional Dependencies
    - For User:
        - Email ----> name, password
    - For Content:
        - contentName, releaseYear ----> contentGenre, poster, studioName
    - For Director
        - directorName ----> age
    - For Actor
        - actorName ----> actorAge
    - For Movies
        - movieName ----> movie_genre, movie_studioname, movie_length
    - For TvSeries
        - tvName ----> no_of_episodes, tvseries_studioname, tvseries_genre

Relations:

- User
    - email → name, password. Since email is a superkey, User is in BCNF.
- Content
    - contentName, releaseYear → contentGenre, poster, studioName.
    - contentName and releaseYear are super key, hence content is in BCNF.
- Director
    - directorName → age
    - DirectorName is key and superkey, Director is in BCNF.
- Actor
    - actorName → actorAge
    - actorName is key and super key, Actor is in BCNF.
- Movies
    - movieName → movie_genre, movie_studioname, movie_length
    - movieName is key and super key. Movie is in BCNF.
- TvSeries
    - tvName → no_of_episodes, tvseries_studioname, tvseries_genre
    - tvName is key and super key. TvSeries is in BCNF.

We can sell all the table has super key at the left hand side and non-prime attribute on right hand side. All functional dependencies fulfill the requirement of BCNF.

Functional Requirement

System Interaction with users

The system provides various functionalities for users. Consumers shall be able to register with users' email and password and use these information to login the system. Consumers shall be able to view different categories of movie list, add the movie to their list and remove a movie from the list. Consumers shall not modify the system movie fix lists, such as the top trending, most popular movie list. The system shall be able to recognize registered or non recognized users. Registered users have the privilege to create their own favorite movie list whereas non registered users shall only be able to view movie list. Consumers can access the system through the web browser on the internet.

Functional Requirement:

Sign Up

- A user shall be able to use their email and password to create a user in our system.
- Once a user enters the information, the system shall be able to create a end-user and enable the user with all the privileges and functions associated with the website.

Login

- A user shall be able to log into the website using correct email and password.
- Once a user logs in, the system shall give access to the user, along with all the privileges and functions associated with the website.

Movies and TV series Listing

- A user shall be able to browse the movies and T.V series from the database provided by the system. The user shall be able to click on the movie in order to enter the subset of the movie.
- The system shall provide a list of movies for user to browse. Each of the movie shall include a brief introduction of the movie.

Display content information

- For a particular content, if the user clicks on the "more info" button, the user shall be able to see a brief description about the content, as well as the actors, directors, genre of content, studio and production company, movie length in minutes (for movies), and number of episodes (for T.V. series).

Rate content

- If the user clicks on the review button for a particular content, the system shall open up a new page, where the system shall provide review writing section and rating dropdown to select a rating (from scale 1 to 10).

Review content

- The user shall be able to write a review comment(400 words max) to a movie.
- When a user clicks on submit, the review and rating should be added into the database.
- The movie, along with the review and the rating, will be stored into user's movie list so that users can have quick access to their movies.

Add a Friend

- The user shall be able to create a connection with other users by entering their email information.
- Once user adds another user, the system shall be able to insert the friend's email information to the user's friend list table.

See Friend's Review List

- A user shall be able to see his or her friend's review list by clicking on the email of a friend from a dropdown menu.

Add to movie list

- A registered user shall be able to add a movie to their movie list by clicking on the submit review button after completing reviewing and rating for a movie.
- The system shall store be able to store a movie to a movie list associated to a user account in order to let the user browse through later.

Delete movie from list

- A user shall be able to delete a desire movie from the movie list. When a user clicks on the delete button the movie is deleted from the database and movie list if automatically reloaded again.

Edit a review and rating

- A user shall be able to edit review and rating
- When a user clicks on edit button, a popup will be shown where the review and the rating will be pre-loaded, and a user shall be able to edit it.
- After clicking on the submit button, the review and rating shall be updated in the database and the page will automatically refreshed to show the updated data.

Non-functional Issues

Graphical User Interface:

The user interface of this application will be user-friendly and easy to use. The search bar for searching movies would be placed on top of the web application for easy accessibility. The features of the application such as movie list would be visible to the user by sorting based on latest, oldest, and rating. There will be dedicated buttons “Invite” which will take the user to his or her friend’s list and allow user to invite other users to watch a movie together. To minimize complexity, the rating will be shown next to the movie title and can also be accessed to edit or change reviews of a movie. The rating will be shown in sets of star shape ranging from 1 to 5. This will increase the accessibility and interaction of user with our web application.

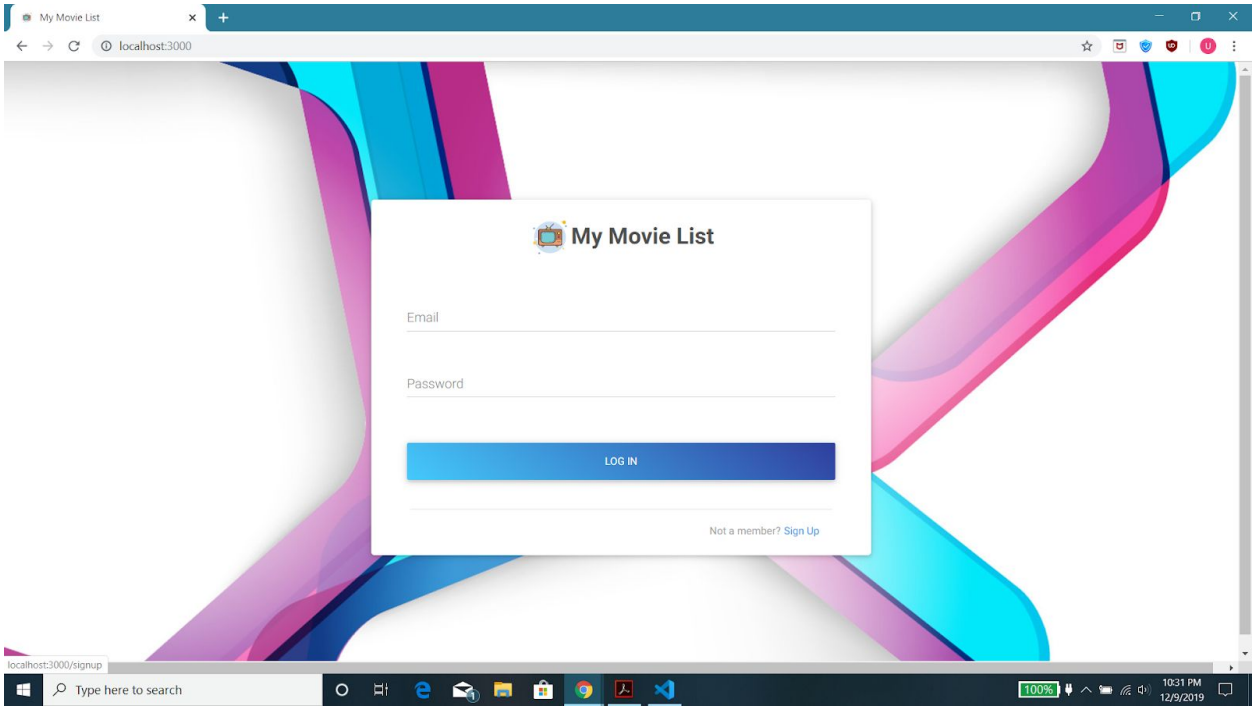
Security:

The user must login with username and password to access his or her movie list. The user must provide a valid email account to create a login credentials for this web application. This prevents others to gather information about the user and maintains confidentiality. The user will have an option to view his or her profile with the other users in the friend’s list. The HTTP server uses TCP/SSL connection which ensures the data will be encrypted, and it will not be easily decrypted by an attacker. TCP also ensures the connection from both ends, server and the client, so the data will be delivered to the client securely without losing the packet throughout the process.

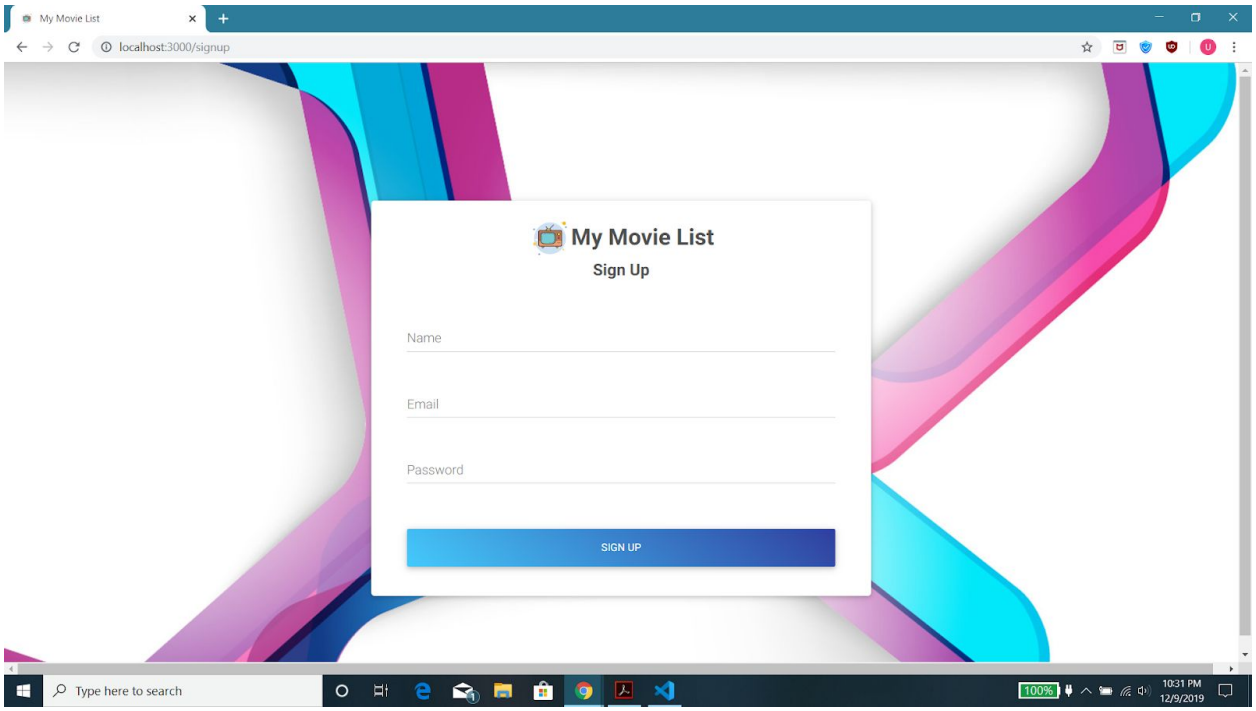
Access Control:

A user will be able to access his or her own movie list, and movies from the database. A user can also share movie list with friends and other users. However, a user can not change or edit other users' movie list. Every user will only have access to their own list, with read and write permission.

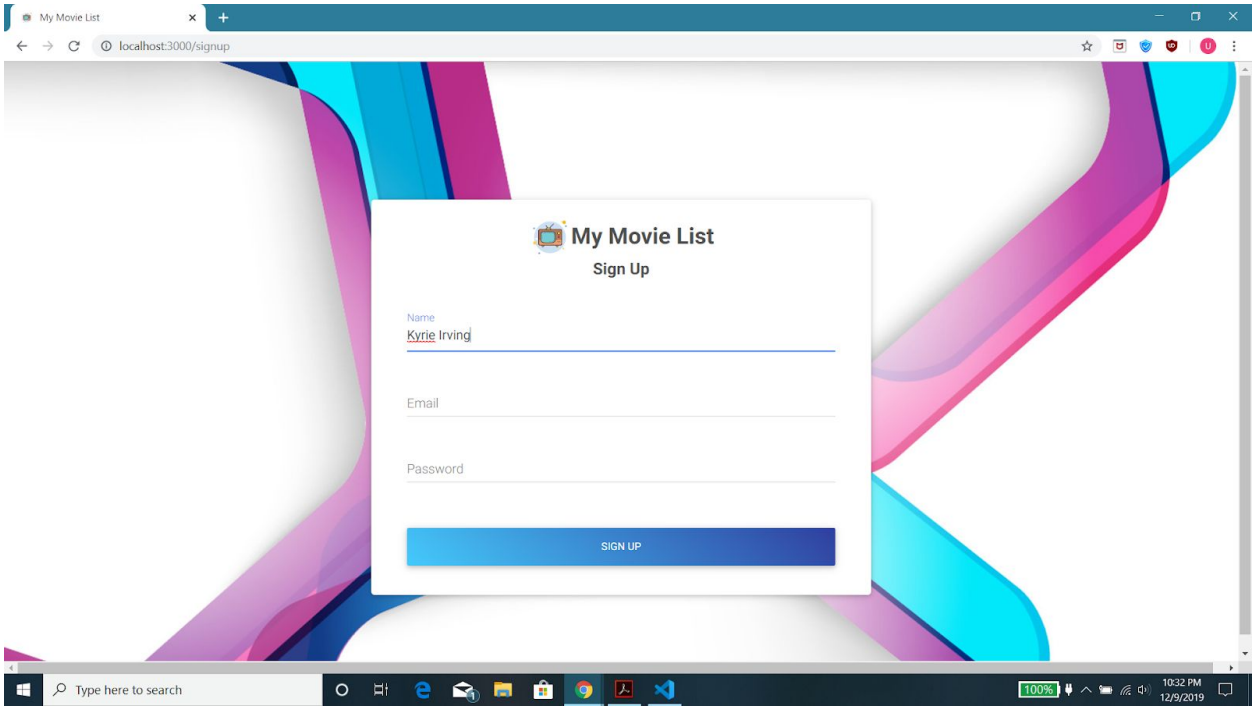
Screenshot of Website:



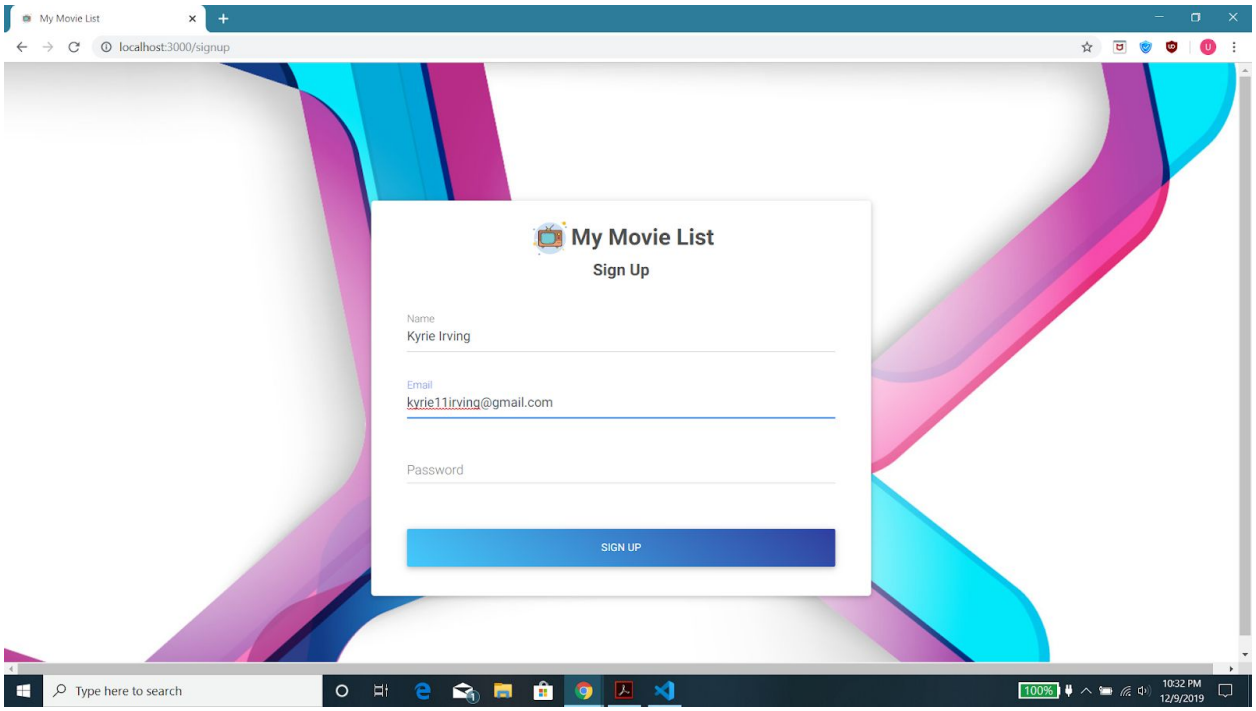
This is the homepage for mymovielist. To register a user, a user must click on the Sign Up link in the bottom right.



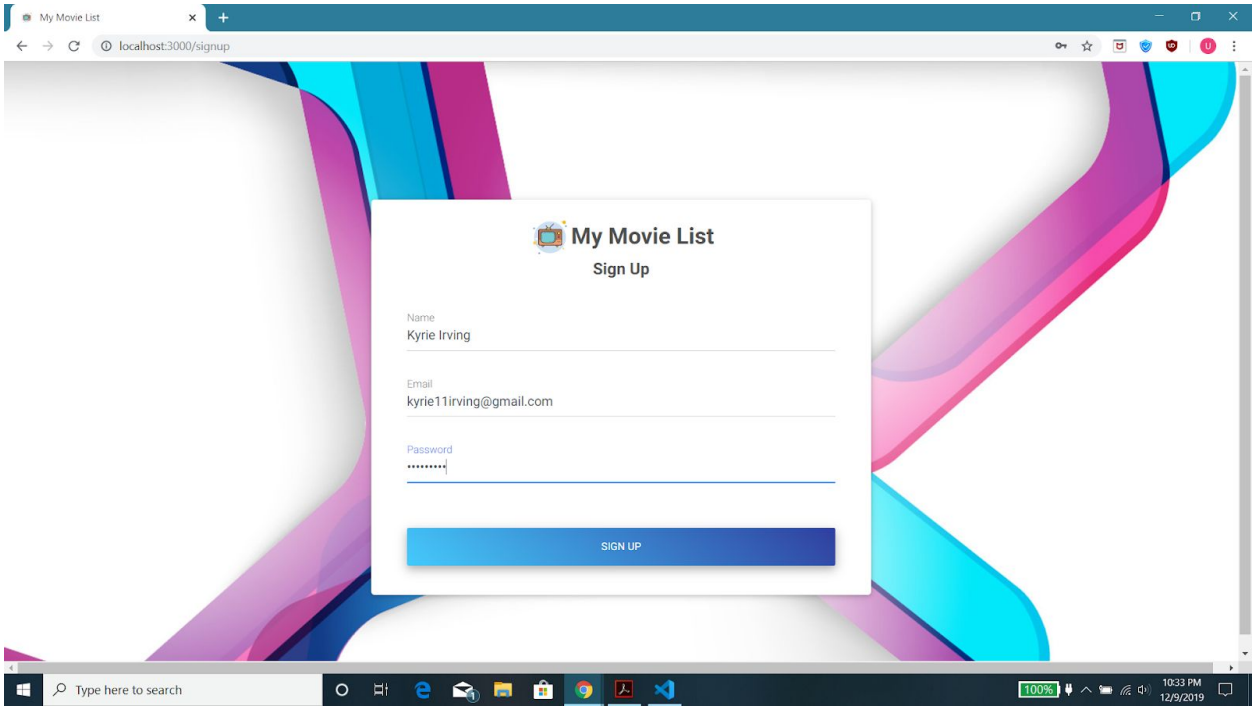
This is the page for Sign Up.



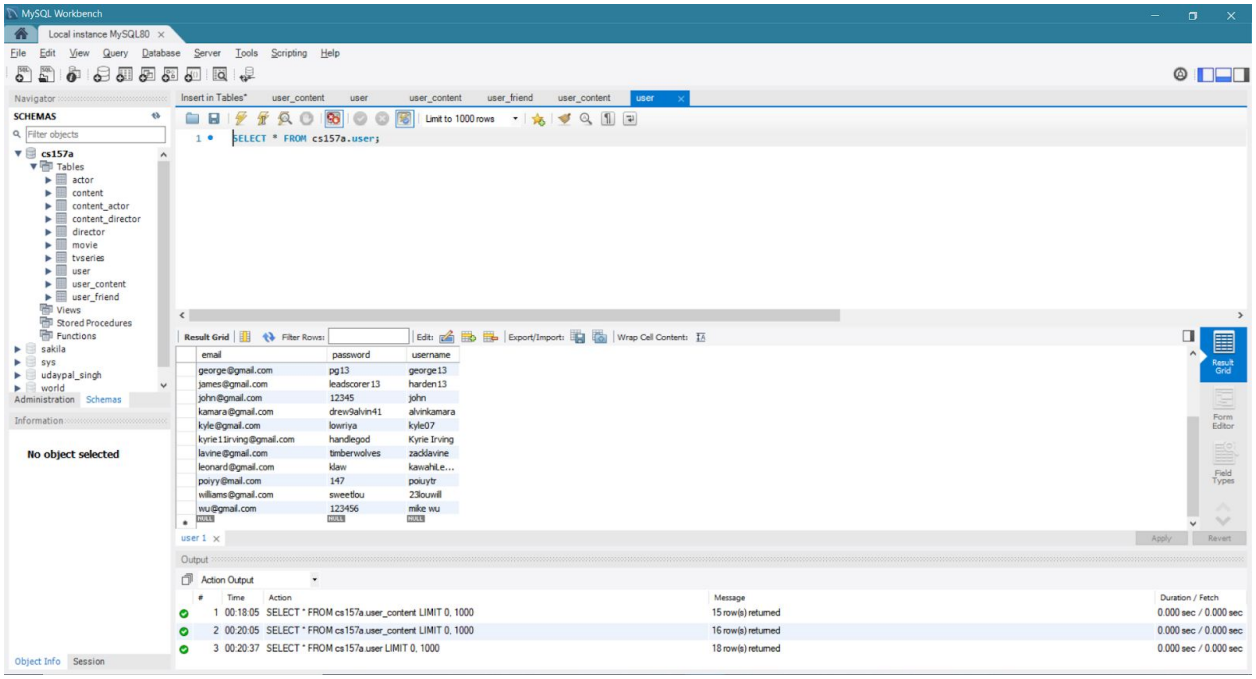
The first input is the name of the user.



Second input of the email of the user.



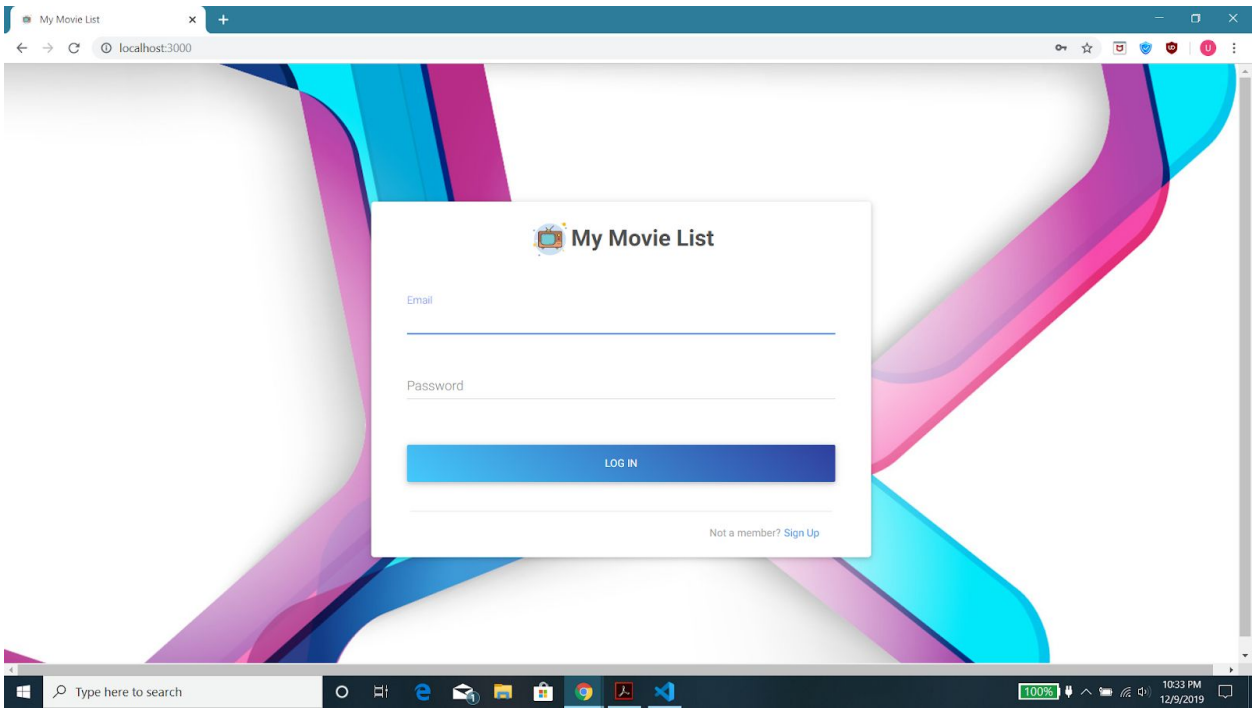
Third input is the password of the user.



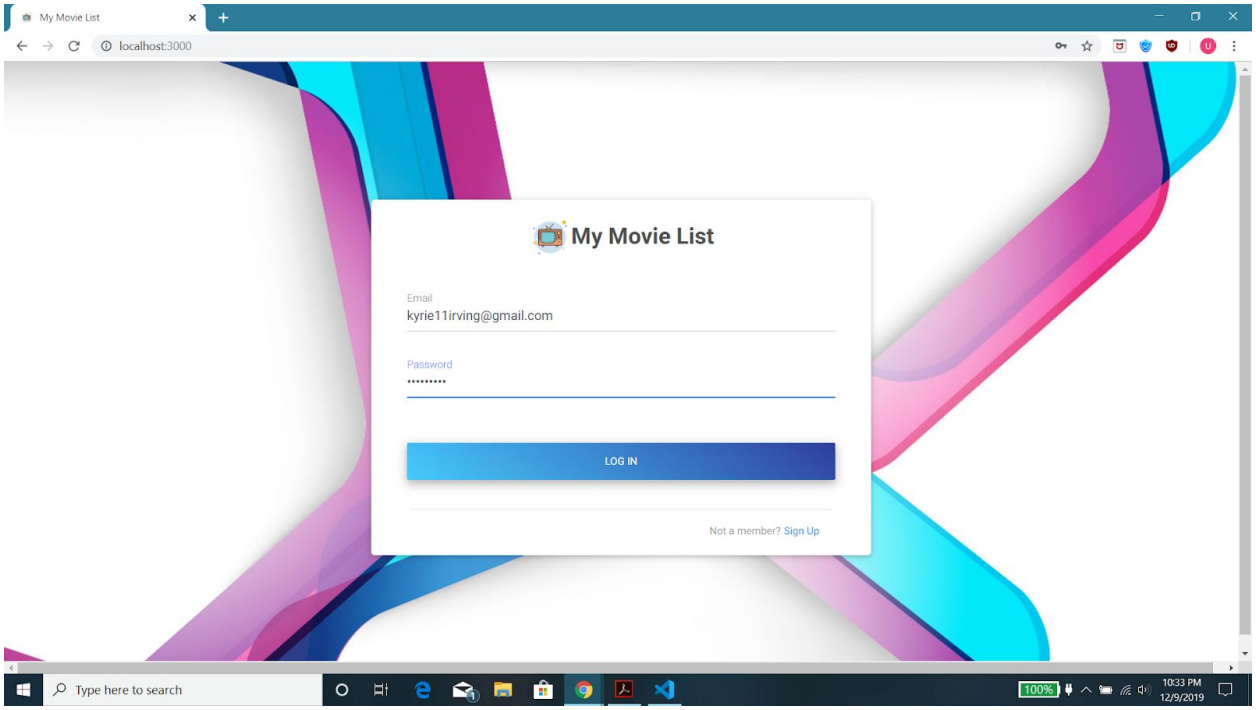
A user is successfully added to the database

```
221  app.get("/signup", (req, res) => {
222    const { email, password, username } = req.query;
223    const INSERT_USER = `INSERT INTO User VALUES('${email}','${password}', '${username}')`;
224    db.query(INSERT_USER, (err, results) => {
225      if (err) {
226        return res.send(err);
227      } else {
228        return res.send("user sucessfully added");
229      }
230    });
231  });
232
```

This query is sent from the backend to add a user to the user table



After clicking on the Sign Up button the user will be redirected to Log In page.



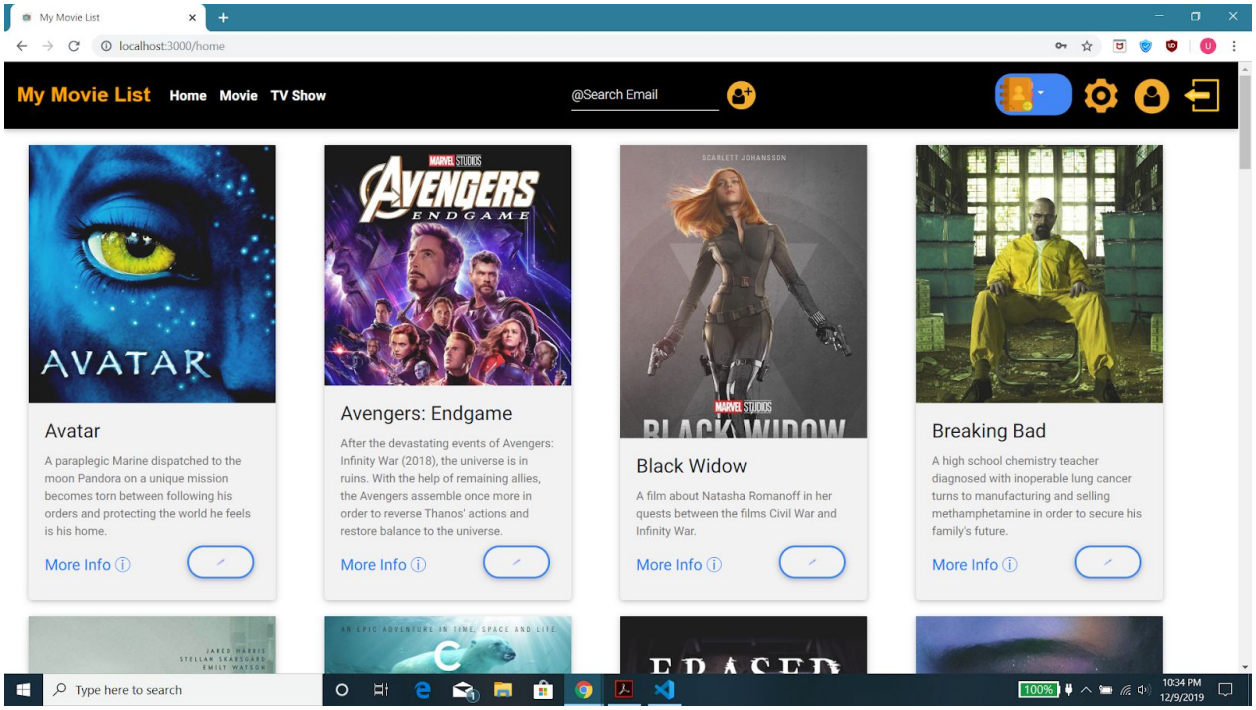
After putting in credentials for a user, the user will be logged into the website.

```
79  app.get("/login", (req, res) => {
80    const { email } = req.query;
81    db.query(`SELECT * FROM User WHERE email = '${email}'`, (err, result) => {
82      if (err) {
83        return res.send(err);
84      } else {
85        return res.json({
86          data: result
87        });
88      }
89    });
```

This query gets the user with the email.

```
116     login = async _ => {
117       const { user } = this.state;
118       await fetch(`http://localhost:4040/login?email=${user.email}`).then(res =>
119         res.json().then(res =>
120           this.setState(
121             {
122               qResult: res.data
123             },
124             () =>
125               this.state.qResult.map(db =>
126                 this.setState({
127                   email: db.email,
128                   password: db.password,
129                   name: db.username
130                 })
131               )
132           )
133         )
134       );
135       if (this.state.password === this.state.user.password) {
136         this.setState({ auth: true });
137       }
138     };
139   }
140
```

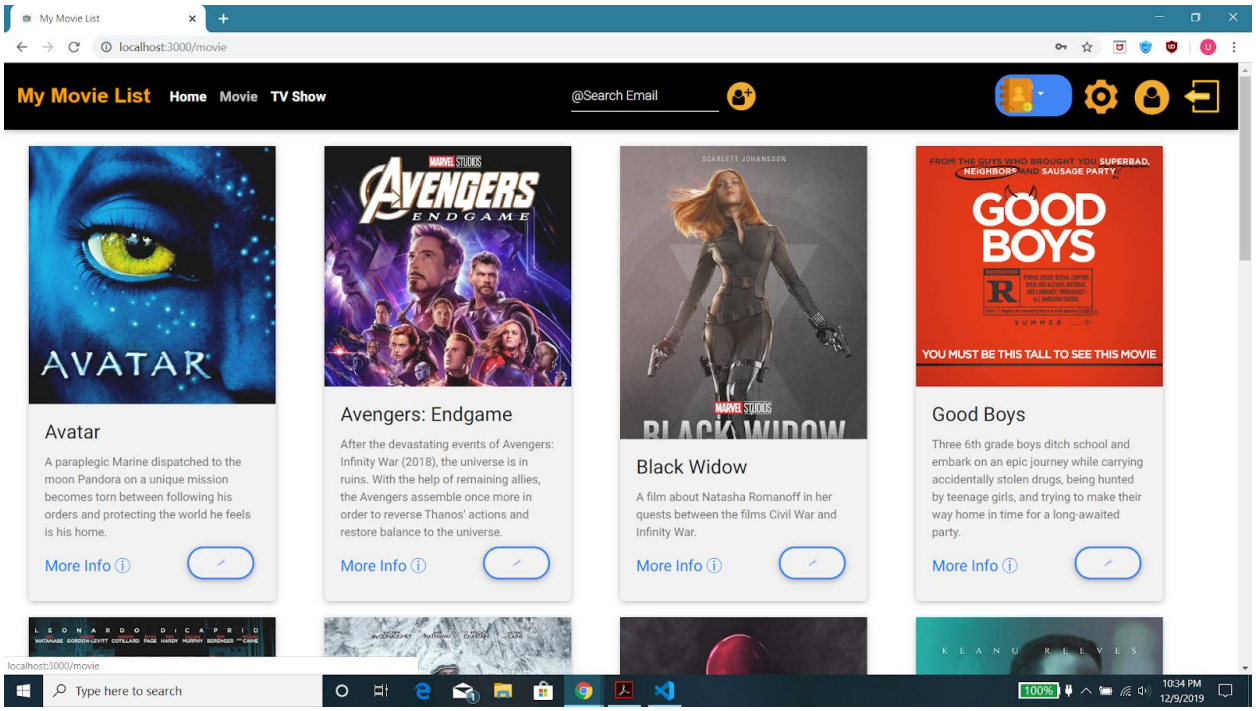
The frontend check the credentials of the user and authorizes the user to log into the website.



This is the homepage of the website with all the movies and tv series.

```
66
67  app.get("/content", (req, res) => {
68    db.query("SELECT * FROM CONTENT", (err, results) => {
69      if (err) {
70        return res.send(err);
71      } else {
72        return res.json({
73          data: results
74        });
75      }
76    });
77  });
78
```

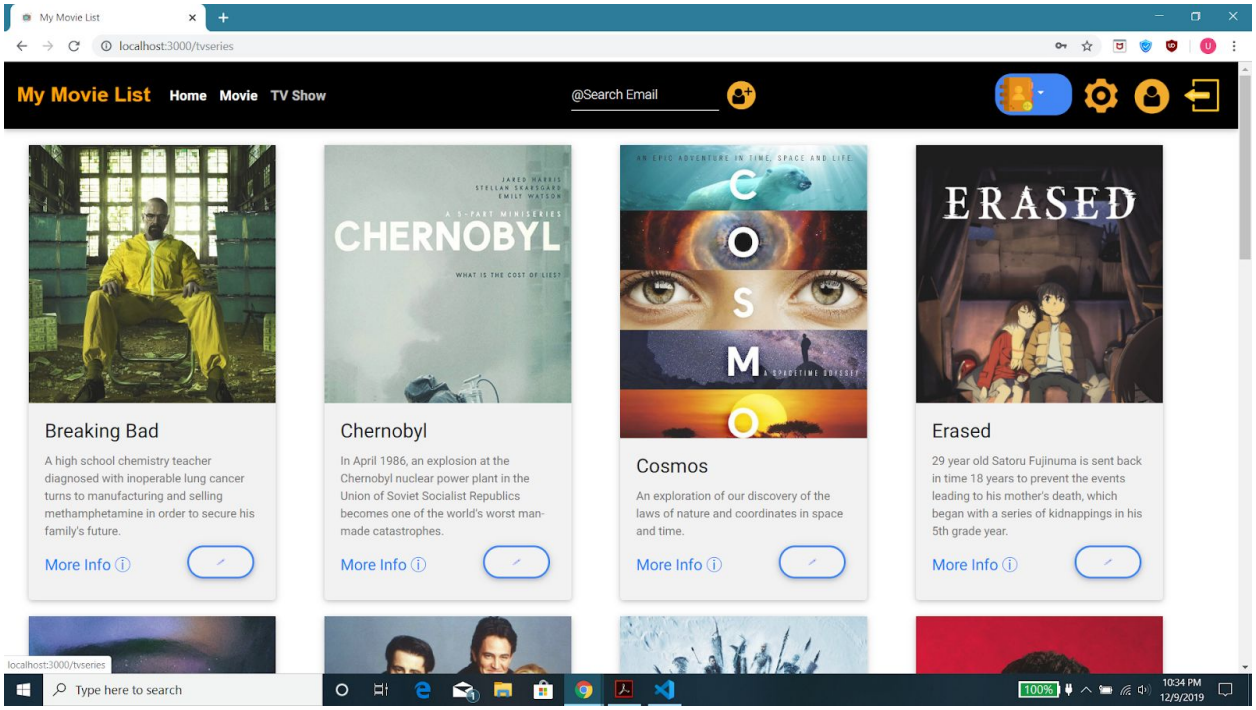
This query gets the content from content entity set.



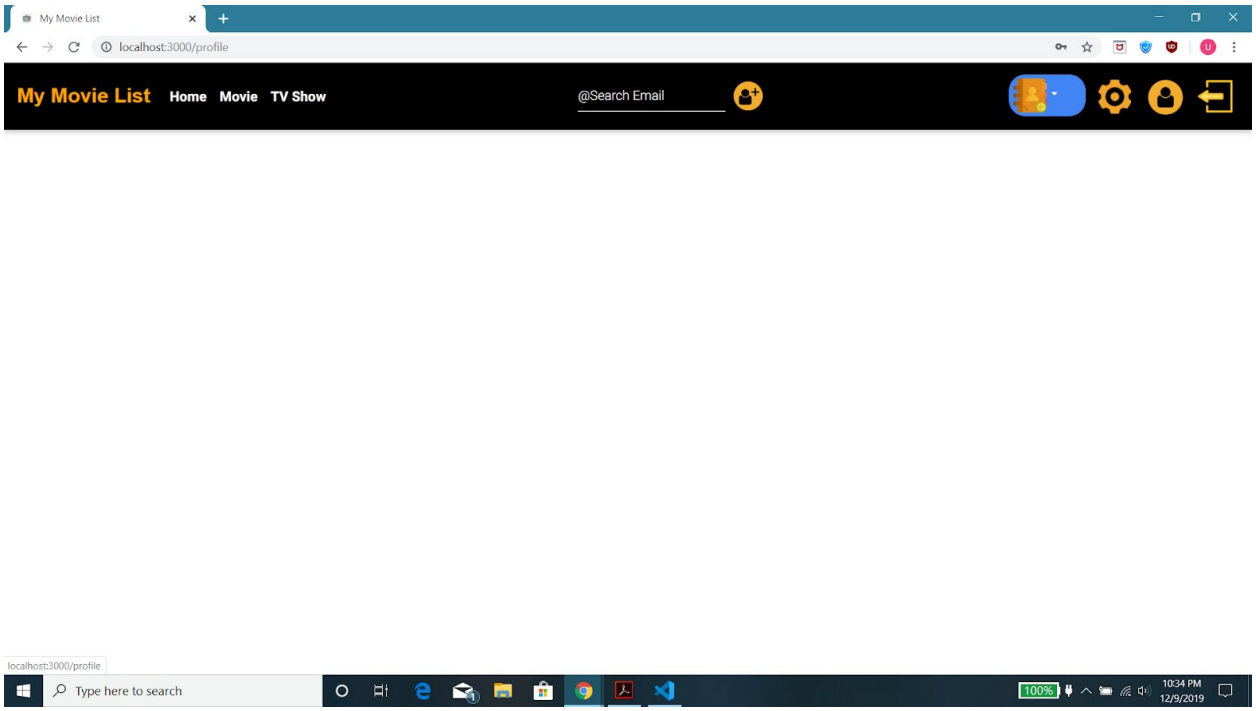
A user can click on “Movie” on the Navbar to list all the movies.

```
206  app.get("/getMovie", (req, res) => {
207    const { movie_contentname, movie_releaseyear } = req.query;
208    const GET_CONTENT = `SELECT * FROM movie WHERE movie_contentname = '${movie_contentname}' AND movie_releaseyear = '${movie_releaseyear}'`;
209
210    db.query(GET_CONTENT, (err, results) => {
211      if (err) {
212        return res.send(err);
213      } else {
214        return res.json({
215          data: results
216        });
217      }
218    });
219  });
220
```

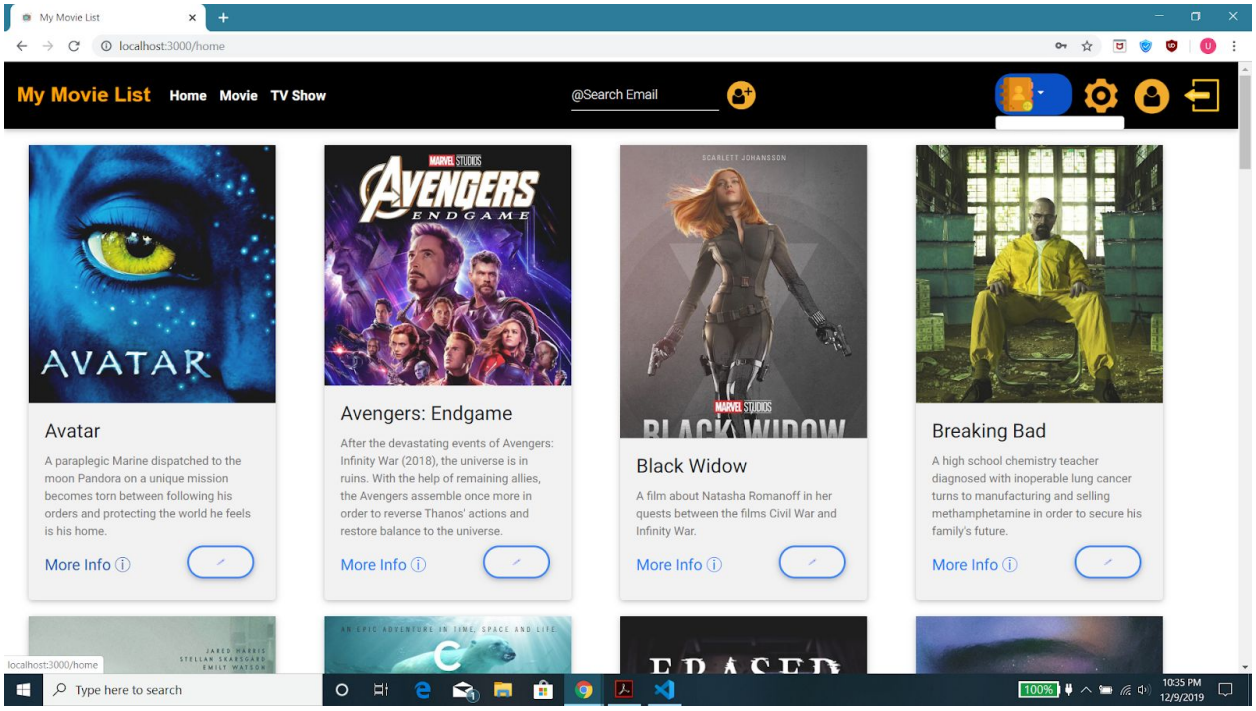
This query gets all the movies from the database.



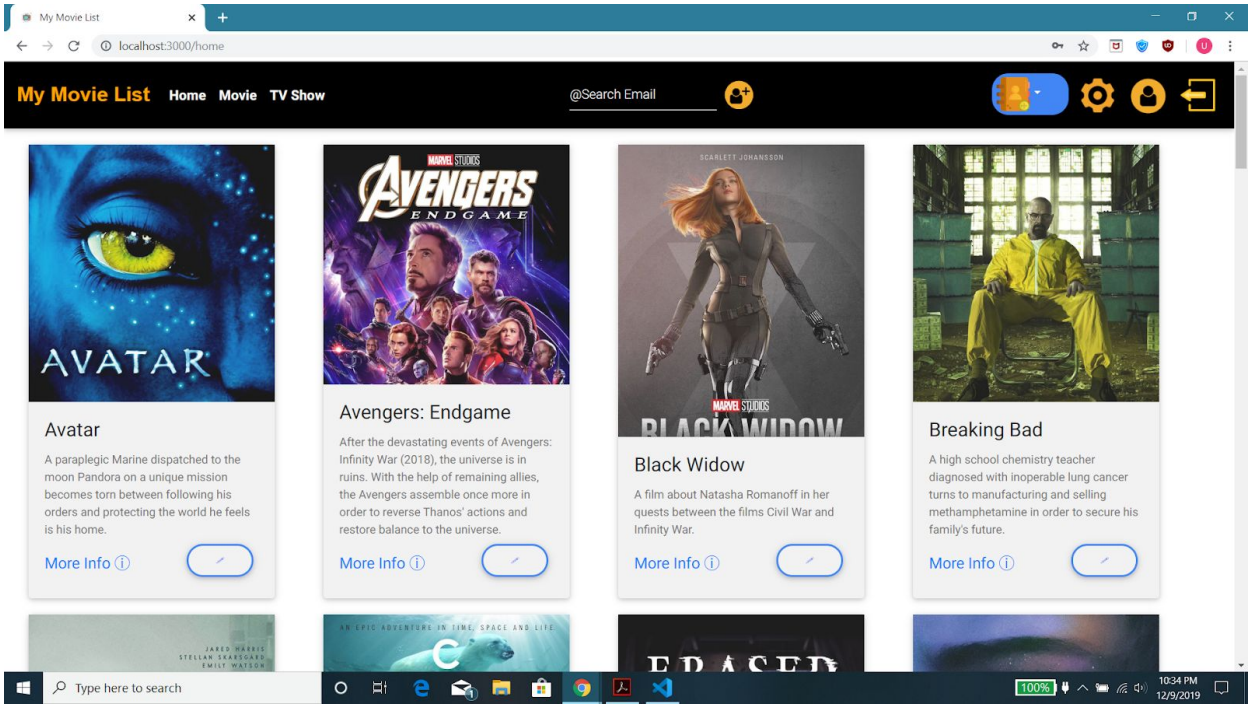
A user can click on “TV Show” in the Navbar to list all the tv series.



When a user clicks on the profile icon on the right side of the navbar, a user can see all the reviews given by a user. Right now, since we just created a user, we don't have any reviews yet.



When you click on the friends list dropdown menu you can see your friends list, you can see your friends

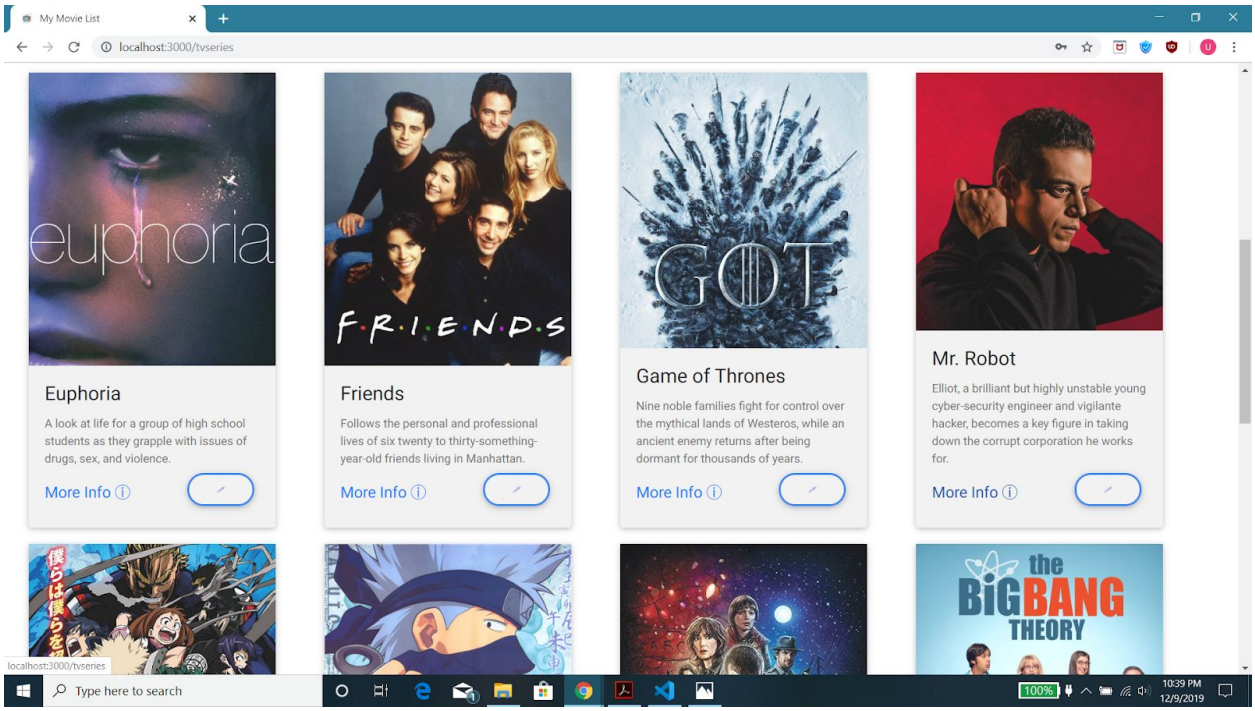


From homepage, a user can click on “More Info” on the movie card, and that redirects the user to more info page.

More info page gives you a detailed summary of the movie, along with genre, studio, length, directors, and actors.

```
app.get("/getContentUnionMovie", (req, res) => {
  const { contentname, releaseyear } = req.query;
  console.log(contentname);
  const GET_CONTENT = `SELECT content.contentname, content.releaseyear, contentgenre, studioname, poster, movie_length, directorname, description
    FROM content, content_director JOIN movie ON contentname = movie_contentname AND releaseyear = movie_releaseyear
    WHERE content.contentname = content_director.contentname AND content.contentname = '${contentname}' AND content.releaseyear = '${releaseyear}'`;
  db.query(GET_CONTENT, (err, results) => {
    if (err) {
      return res.send(err);
    } else {
      return res.json({
        data: results
      });
    }
  });
});
```

This query returns all the information for movie content.

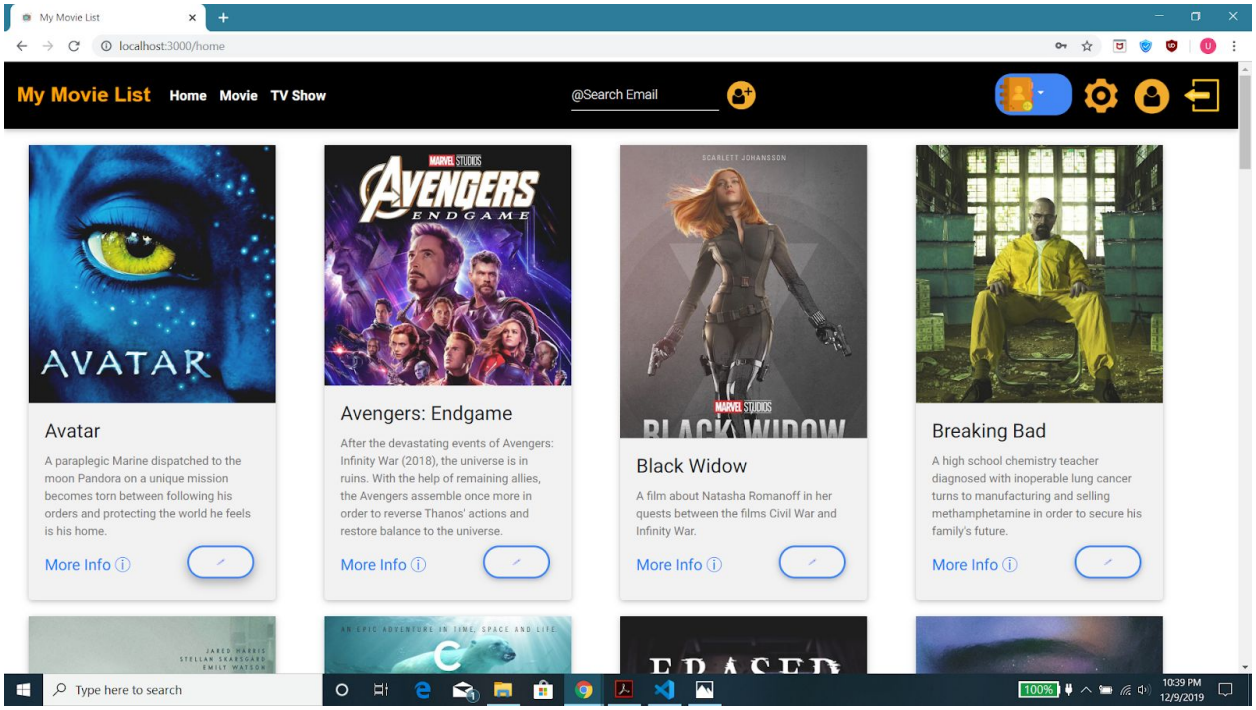


From tv series, you can also click on more info, and it gives more detail about a tv series.

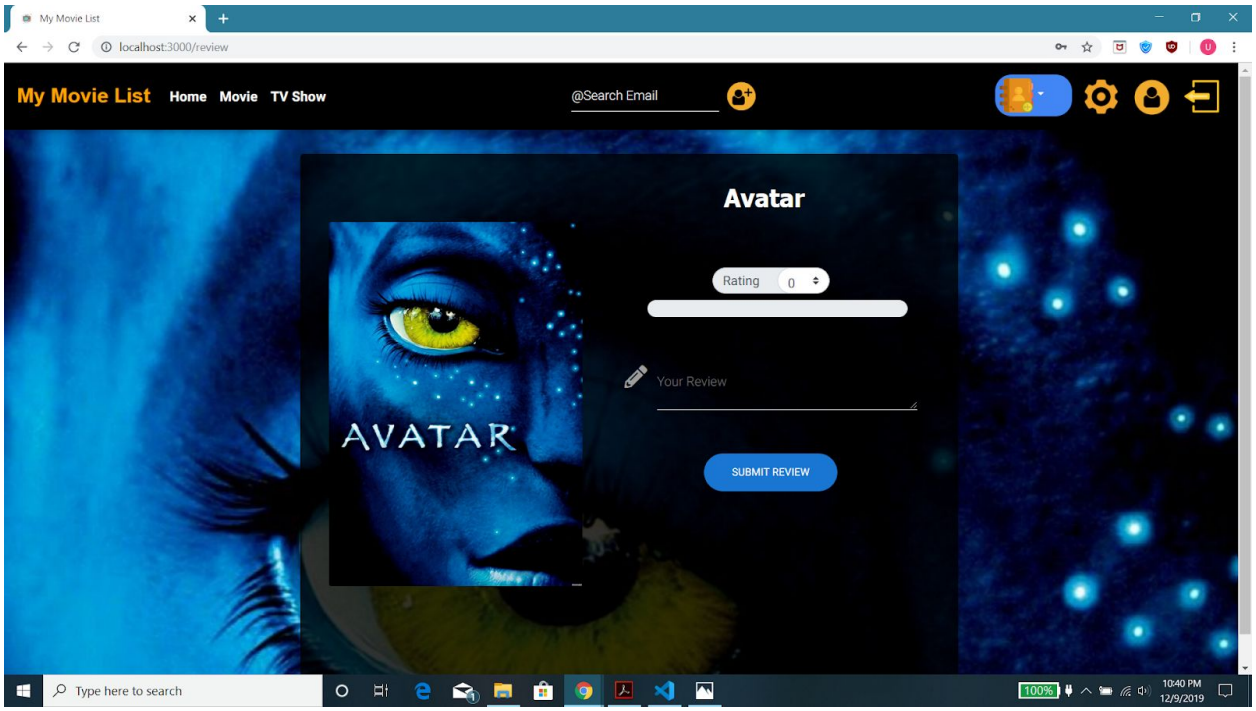
“More info” of a tv series gives you Episodes, instead of movie length.

```
app.get("/getContentUnionTvSeries", (req, res) => {
  const { contentname, releaseyear } = req.query;
  const GET_CONTENT = `SELECT content.contentname, content.releaseyear, contentgenre, studioname, poster, no_of_episodes, directorname, description
    FROM content, content_director JOIN tvseries ON contentname = tvseries_contentname AND releaseyear = tvseries_releaseyear
    WHERE content.contentname = content_director.contentname AND content.contentname = '${contentname}' AND content.releaseyear = '${releaseyear}'`;
  db.query(GET_CONTENT, (err, results) => {
    if (err) {
      return res.send(err);
    } else {
      return res.json({
        data: results
      });
    }
  });
});
```

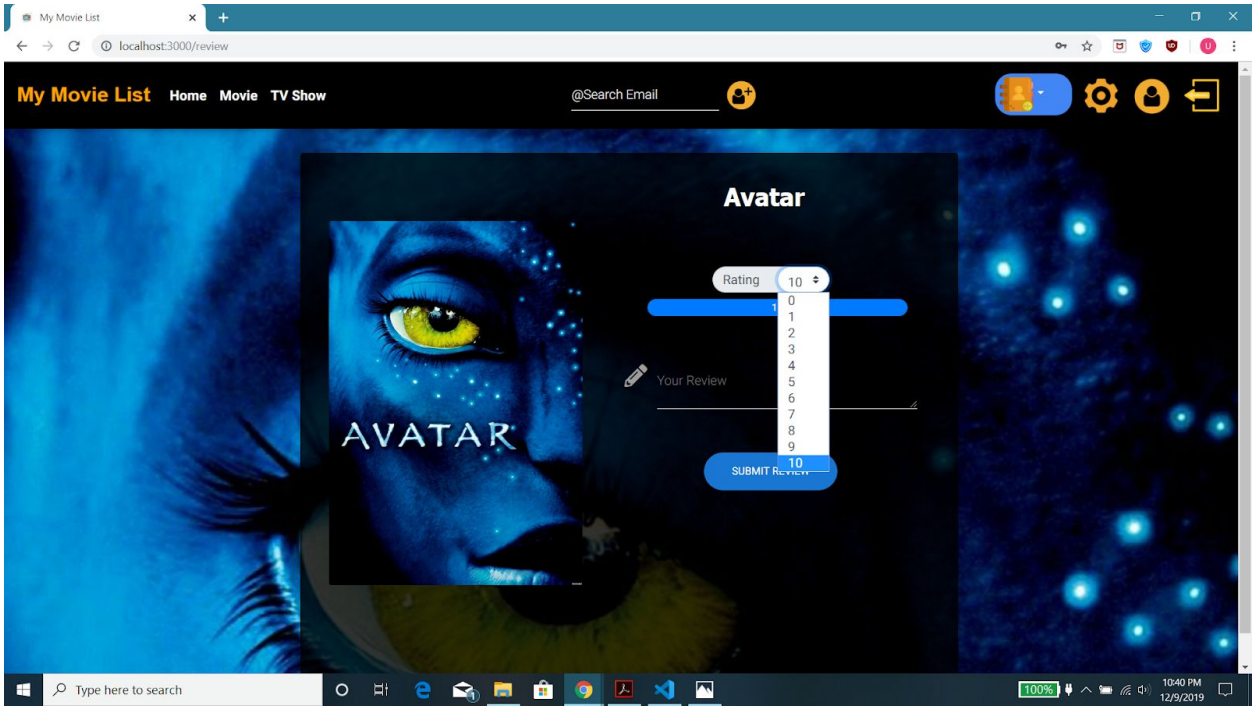
This returns the data for tv show content



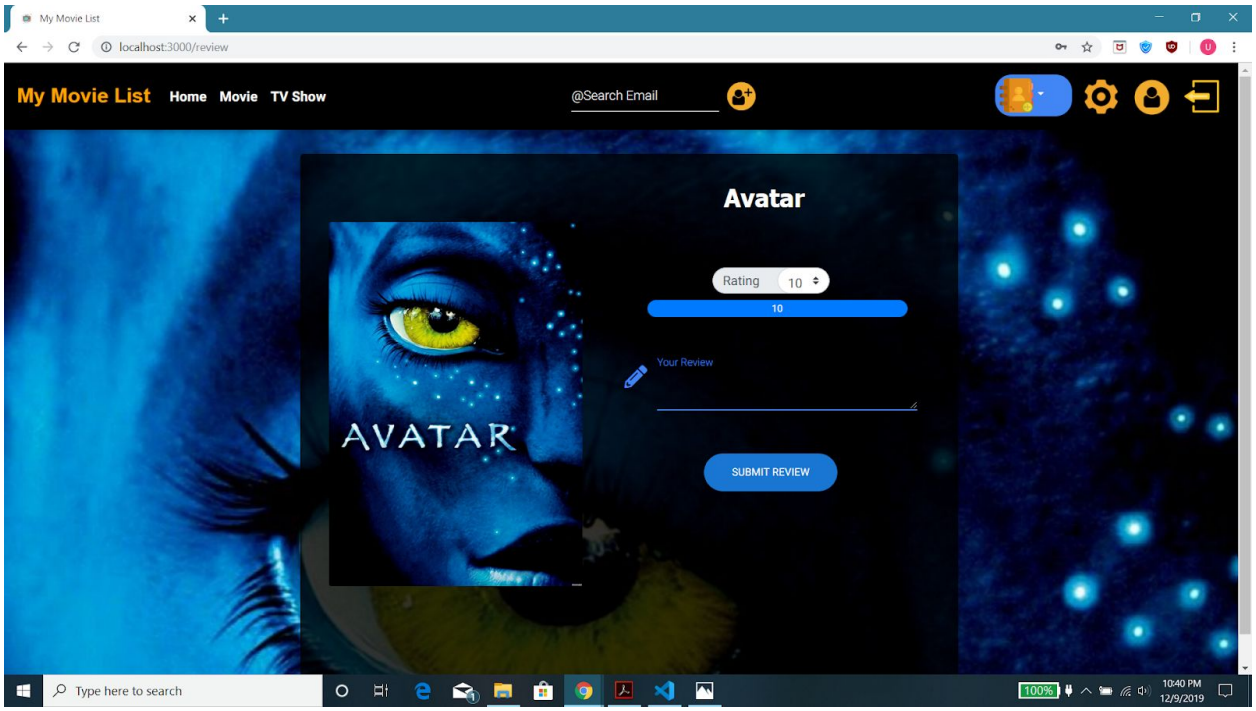
From homepage, a user can also click on the review button, and it will take a user to review page



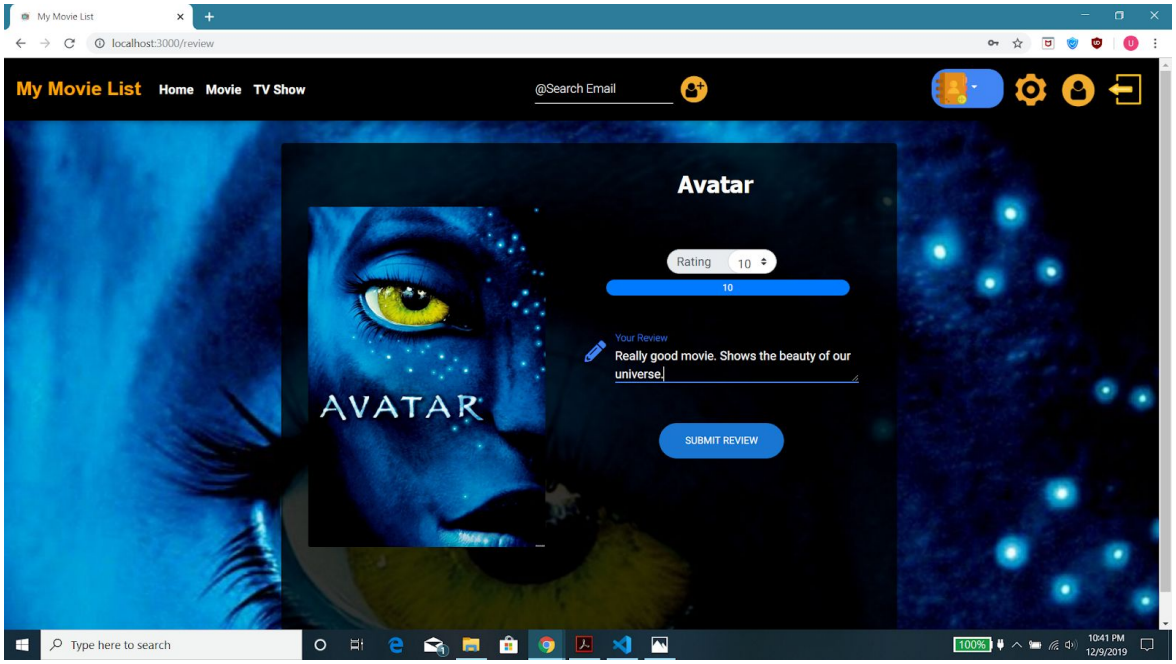
A review page gives you two inputs: rating and review.



From the drop down a user can rate a movie from a scale (1 to 10).



The review bar progresses according to the rating given by a user.



A user can write a review for a movie, and once a user click on submit the review is added to the backend MySql, and the user is redirected to the homepage.

```
app.get("/addReview", (req, res) => {
  const { email, contentname, releaseyear, review, rating } = req.query;
  const INSERT_USER_CONTENT = `INSERT INTO user_content VALUES('${email}','${contentname}','${releaseyear}','${review}','${rating}')`;
  db.query(INSERT_USER_CONTENT, (err, results) => {
    if (err) {
      return res.send(err);
    } else {
      return res.send("review sucessfullt added");
    }
  });
});

```

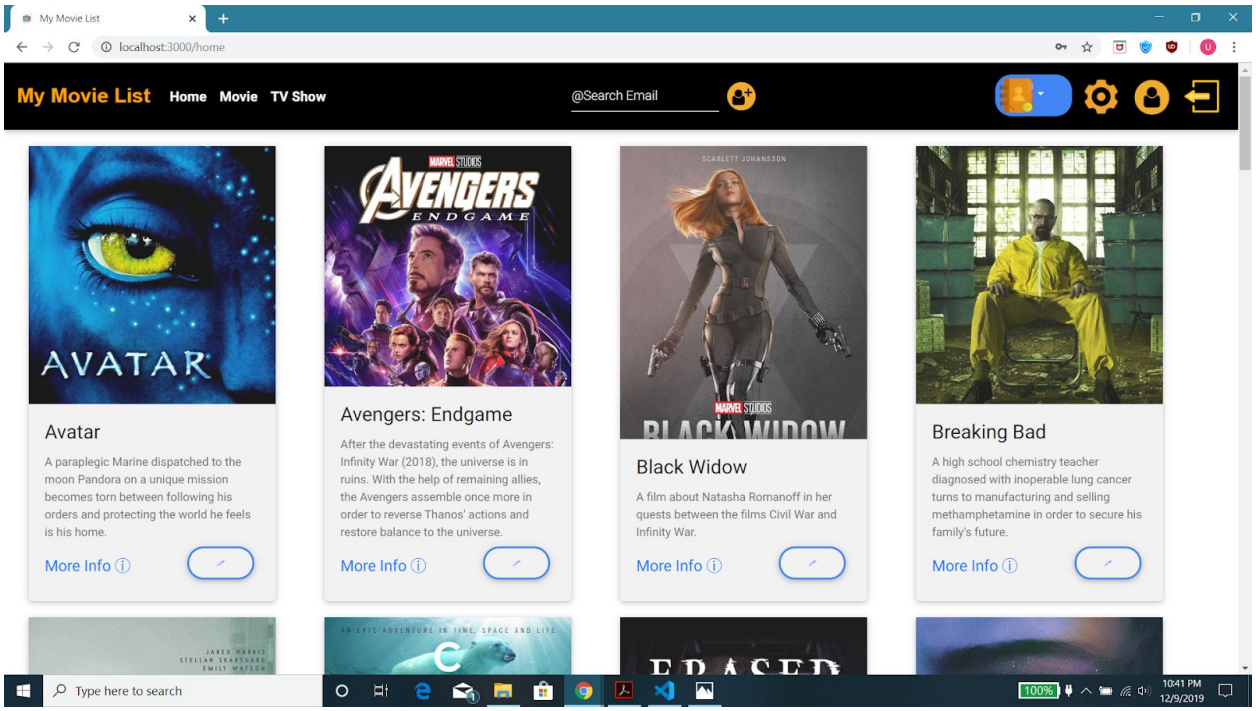
This query adds review to the backend for a particular user.

```
1 •  SELECT * FROM cs157a.user_content;
```

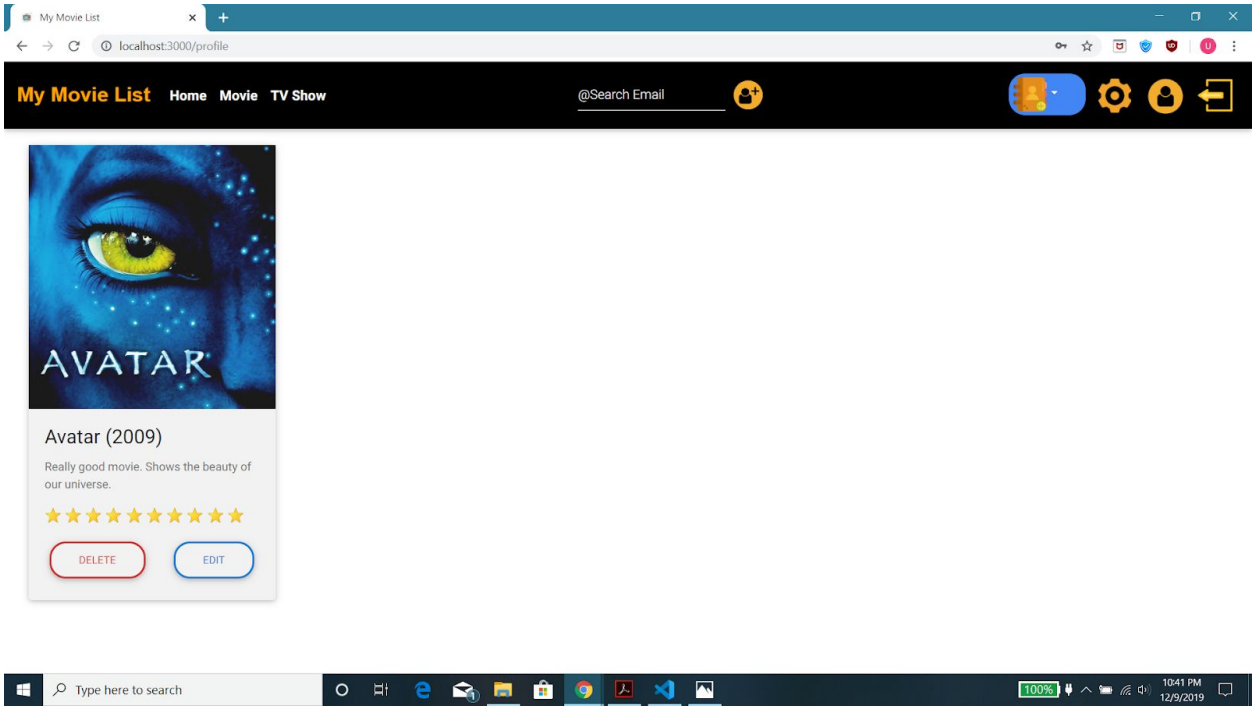
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

email	contentname	releaseyear	review	rating
ben@gmail.com	Cosmos	2014	Tyson makes it the best. But OG Cosmos by Carl Sagan is legendary	10
ben@gmail.com	It	2017	Great jump scare. Stephen King is a king of horror movies.	7
ben@gmail.com	John Wick	2014	Keanu Reeves is a legend, period.	5
ben@gmail.com	John Wick: Chapter 3 – Parabellum	2019	A legend never dies.	9
ben@gmail.com	Ready Player One	2018	Liked the concept of VR games coming to real life.	8
bob@gmail.com	Naruto	2002	one of the best anime of all time	10
carmelo.anthony@gmail.com	The Big Bang Theory	2007	Hilarious show. Gets funnier every time I watch it. New seasons are kinda slow tho.	7
carmelo.anthony@gmail.com	The Boys	2019	It was a good show. Finished in one day.	10
george@gmail.com	Erased	2016	This anime is a master piece. Waiting for the next season.	9
george@gmail.com	Inception	2010	I watched it many times. This movie is so well written. The story line is incredible.	8
george@gmail.com	Mr. Robot	2015	Inspires me every day to become a hacker. Solid 10/10 show	10
john@gmail.com	Avengers: Endgame	2019	Very good movie.	8
john@gmail.com	Stranger Things	2016	Great thriller. I like demogorgons.	10
kyrie11irving@gmail.com	Avatar	2009	Really good movie. Show the beauty of our universe.	10
leonard@gmail.com	Spider-Man: Far from Home	2019	The scenery is incredible. Like the direction where marvel phase 4 is headed.	10
poiyy@mail.com	Avengers: Endgame	2019	Deserves 10 but my favorite avenger dies so I am giving it a 9. Overall, a great movie.	9

As can be seen a review is added to the backend under the email of “kyrie11irving@gmail.com”, contentname as “Avatar” and releaseyear “2009”.



After redirecting to the homepage, user can visit his or her profile page to see reviewed content.

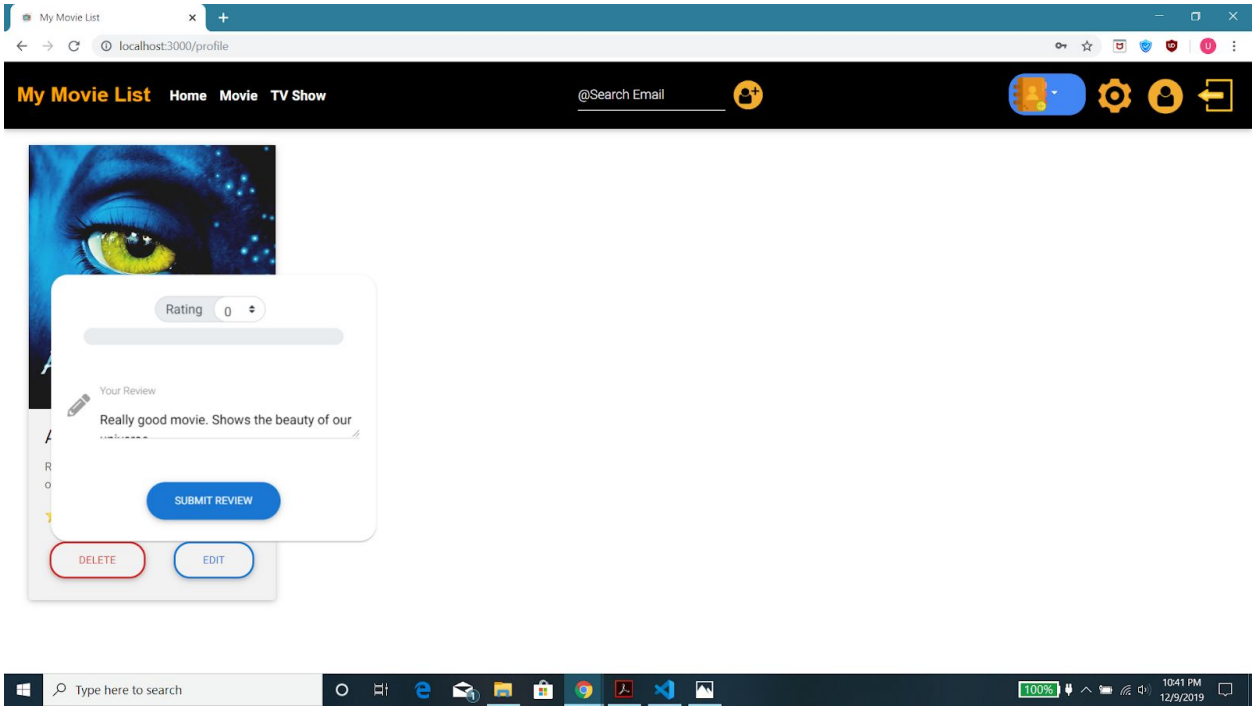


After click on the profile page a user can see the review and rating given by him or her under profile.

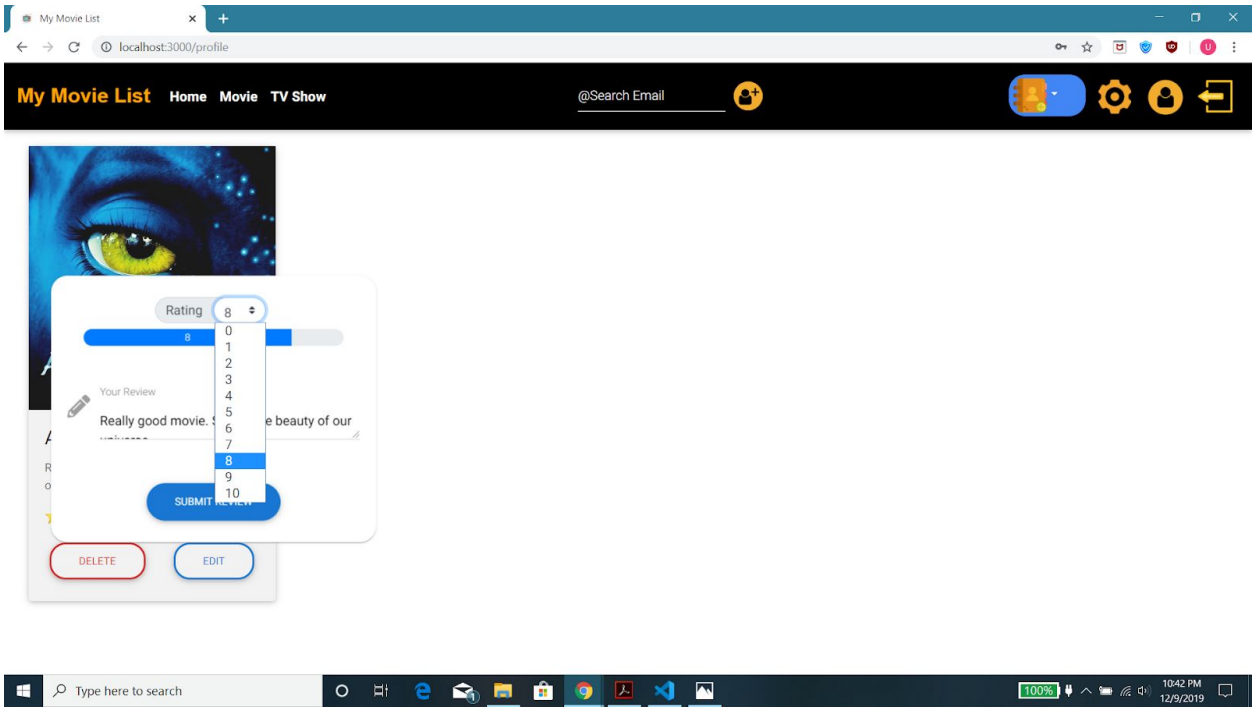
```
app.get('/getReview', (req, res)=>{
  const{email} = req.query;
  const GET_CONTENT =`SELECT content.contentname, content.releaseyear, content.poster, review, rating
  FROM content JOIN user_content ON content.contentname = user_content.contentname AND content.releaseyear = user_content.releaseyear
  WHERE user_content.email = '${email}'`;
  db.query(GET_CONTENT, (err, results)=>{
      if(err){
          return res.send(err);
      }
      else{
          return res.json({
              data: results
          })
      }
  });

})
```

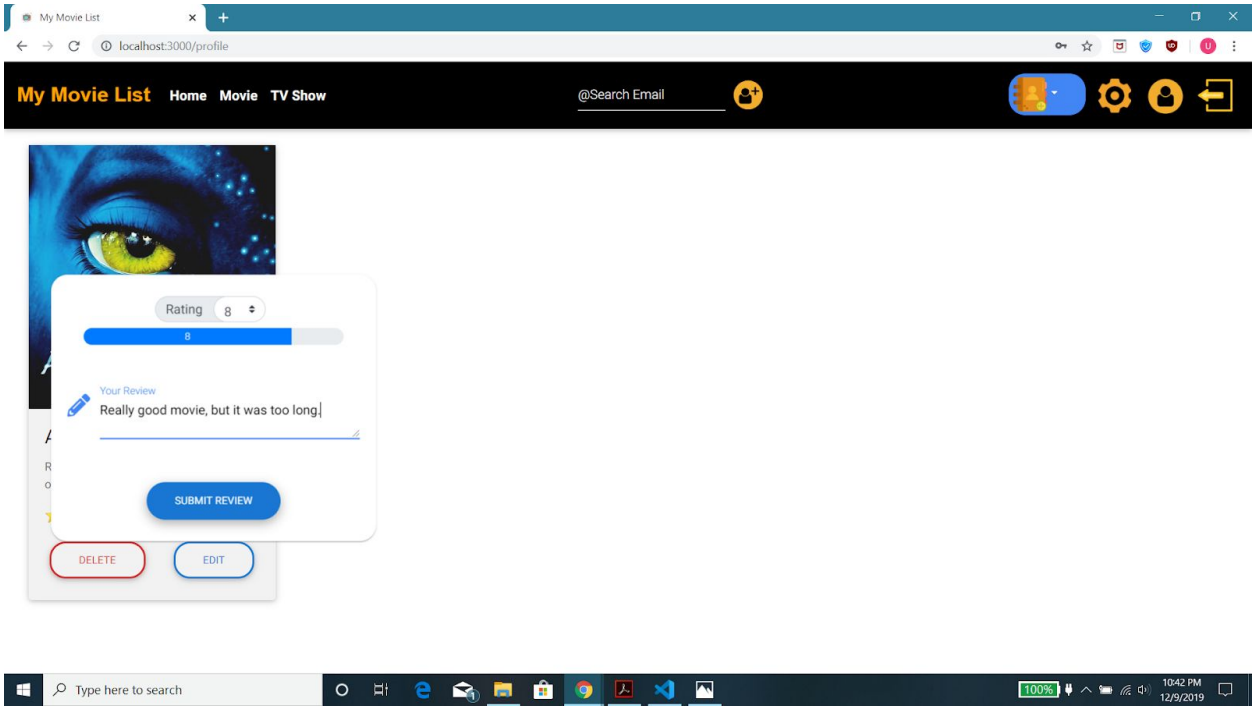
This query returns the list of reviews for a user.



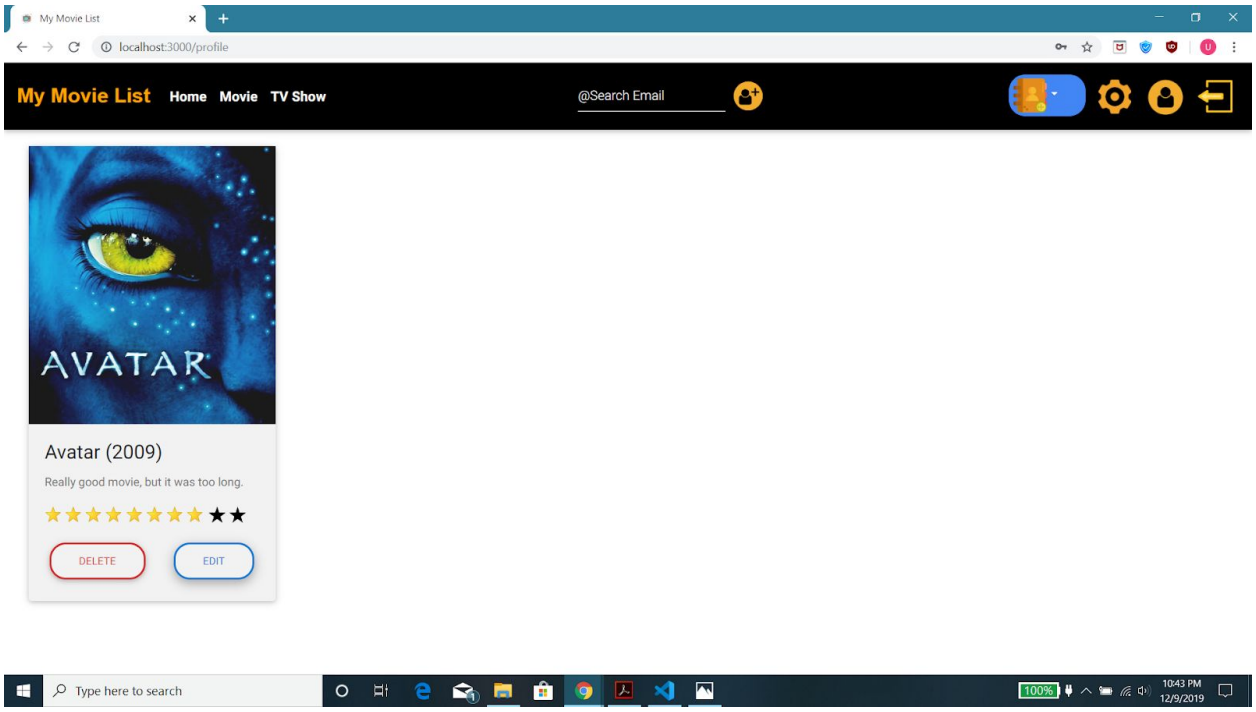
A user is allowed to edit a review by clicking on the review button.



A user can modify the rating by the dropdown scaled (1 to 10).



A user can also edit the review and click on submit.

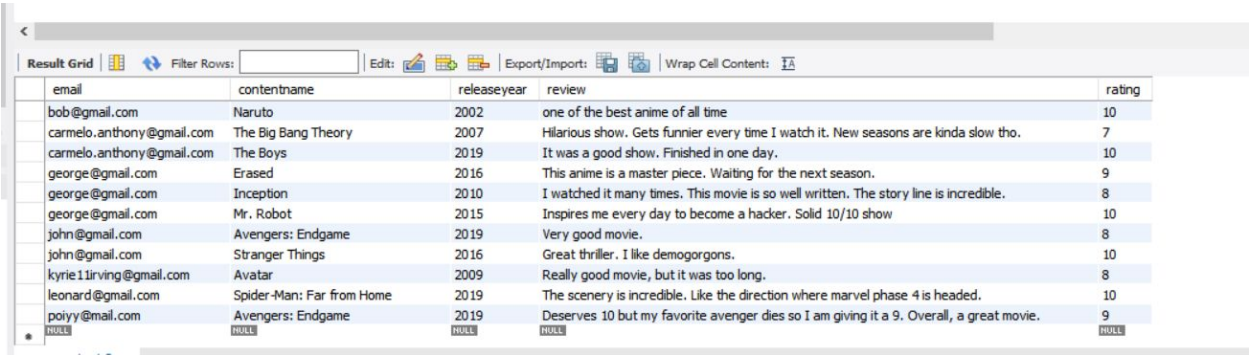


After clicking on submit, the page is automatically refreshed and change the review and rating on the page and the backend.

```
app.get("/deleteReview", (req, res) => {
  const { email, contentname, releaseyear } = req.query;
  const DELETE_REVIEW = `DELETE FROM user_content WHERE email='${email}' AND contentname ='${contentname}' AND releaseyear='${releaseyear}'`;
  db.query(DELETE_REVIEW, (err, results) => {
    if (err) {
      return res.send(err);
    } else {
      return res.send("Review Deleted");
    }
  });
});

app.get("/updateReview", (req, res) => {
  const { email, contentname, releaseyear, review, rating } = req.query;
  const UPDATE_REVIEW =
  `UPDATE user_content SET review='${review}', rating = '${rating}' WHERE email = '${email}' AND contentname = '${contentname}' AND releaseyear = '${releaseyear}'`;
  db.query(UPDATE_REVIEW, (err, results) => {
    if (err) {
      return res.send(err);
    } else {
      return res.send("Successfully Updated");
    }
  });
});
```

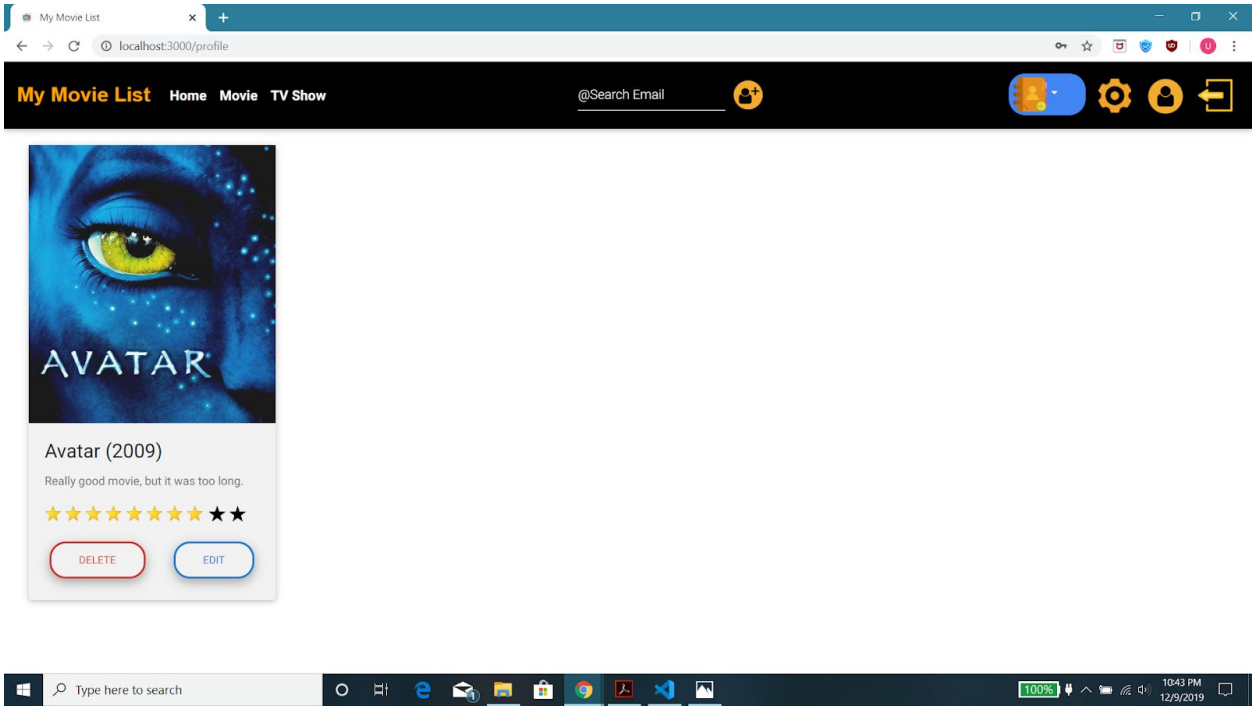
These two queries are used to update or delete a review for a user



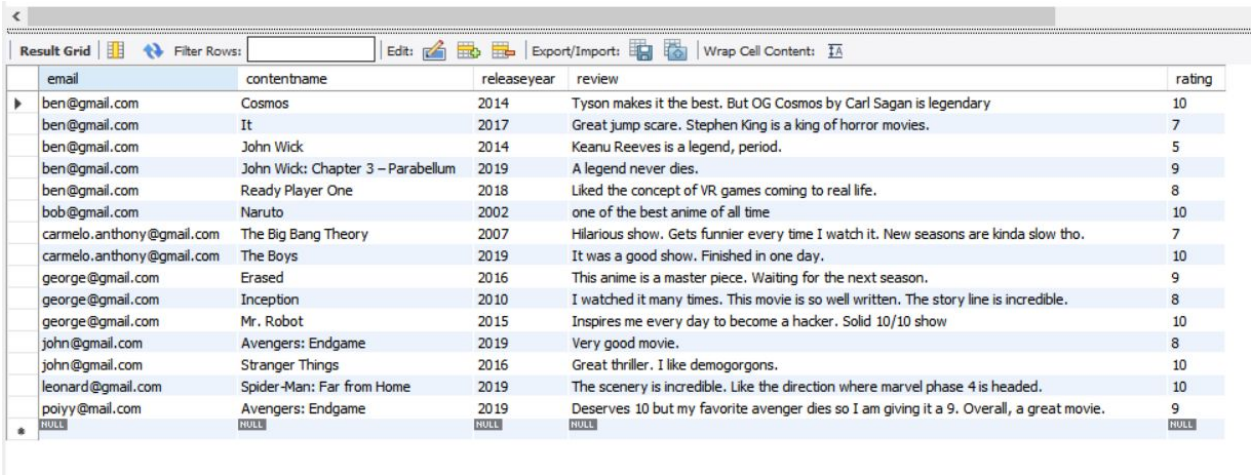
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

email	contentname	releaseyear	review	rating
bob@gmail.com	Naruto	2002	one of the best anime of all time	10
carmelo.anthony@gmail.com	The Big Bang Theory	2007	Hilarious show. Gets funnier every time I watch it. New seasons are kinda slow tho.	7
carmelo.anthony@gmail.com	The Boys	2019	It was a good show. Finished in one day.	10
george@gmail.com	Erased	2016	This anime is a master piece. Waiting for the next season.	9
george@gmail.com	Inception	2010	I watched it many times. This movie is so well written. The story line is incredible.	8
george@gmail.com	Mr. Robot	2015	Inspires me every day to become a hacker. Solid 10/10 show	10
john@gmail.com	Avengers: Endgame	2019	Very good movie.	8
john@gmail.com	Stranger Things	2016	Great thriller. I like demogorgons.	10
kyrie11irving@gmail.com	Avatar	2009	Really good movie, but it was too long.	8
leonard@gmail.com	Spider-Man: Far from Home	2019	The scenery is incredible. Like the direction where marvel phase 4 is headed.	10
poiyy@gmail.com	Avengers: Endgame	2019	Deserves 10 but my favorite avenger dies so I am giving it a 9. Overall, a great movie.	9
NULL	NULL	NULL	NULL	NULL

As can be seen in the picture the value for review and rating is updated for user kyrie11irving@gmail.com.



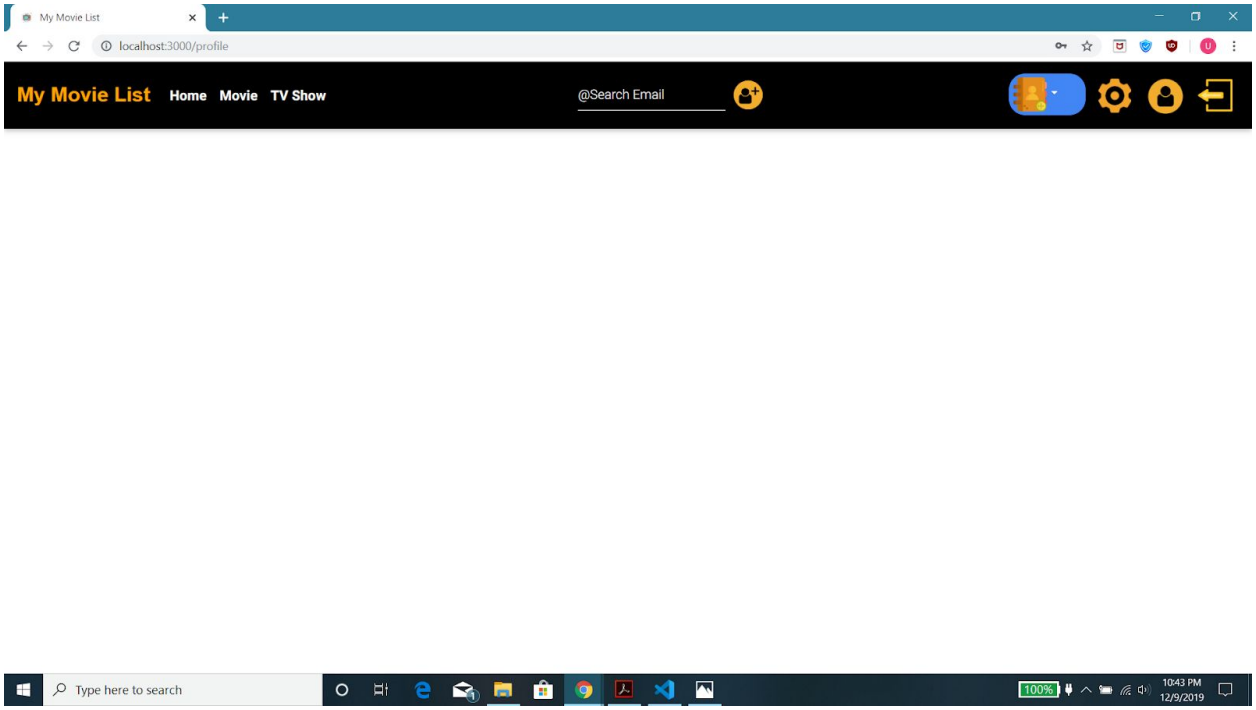
When you click on the delete button, it deletes the movie from a user's list instantly.



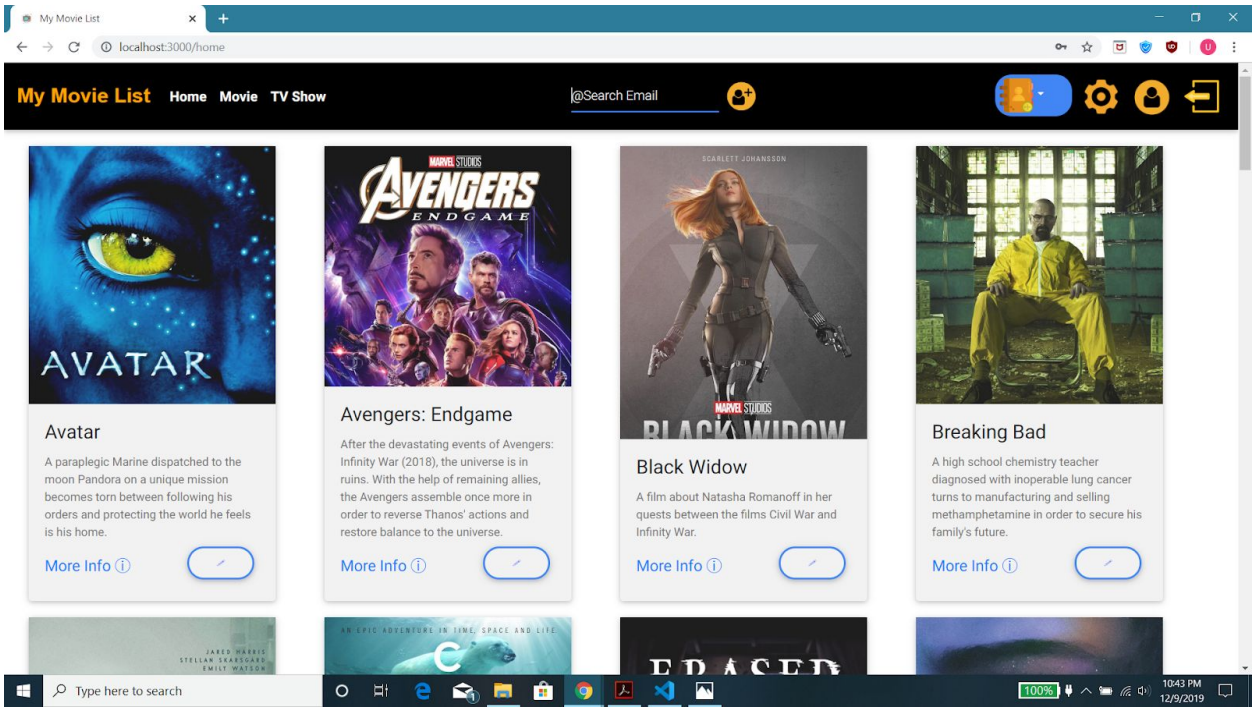
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

email	contentname	releaseyear	review	rating
ben@gmail.com	Cosmos	2014	Tyson makes it the best. But OG Cosmos by Carl Sagan is legendary	10
ben@gmail.com	It	2017	Great jump scare. Stephen King is a king of horror movies.	7
ben@gmail.com	John Wick	2014	Keanu Reeves is a legend, period.	5
ben@gmail.com	John Wick: Chapter 3 – Parabellum	2019	A legend never dies.	9
ben@gmail.com	Ready Player One	2018	Liked the concept of VR games coming to real life.	8
bob@gmail.com	Naruto	2002	one of the best anime of all time	10
carmelo.anthony@gmail.com	The Big Bang Theory	2007	Hilarious show. Gets funnier every time I watch it. New seasons are kinda slow tho.	7
carmelo.anthony@gmail.com	The Boys	2019	It was a good show. Finished in one day.	10
george@gmail.com	Erased	2016	This anime is a master piece. Waiting for the next season.	9
george@gmail.com	Inception	2010	I watched it many times. This movie is so well written. The story line is incredible.	8
george@gmail.com	Mr. Robot	2015	Inspires me every day to become a hacker. Solid 10/10 show	10
john@gmail.com	Avengers: Endgame	2019	Very good movie.	8
john@gmail.com	Stranger Things	2016	Great thriller. I like demogorgons.	10
leonard@gmail.com	Spider-Man: Far from Home	2019	The scenery is incredible. Like the direction where marvel phase 4 is headed.	10
poiyy@mail.com	Avengers: Endgame	2019	Deserves 10 but my favorite avenger dies so I am giving it a 9. Overall, a great movie.	9
NULL	NULL	NULL	NULL	NULL

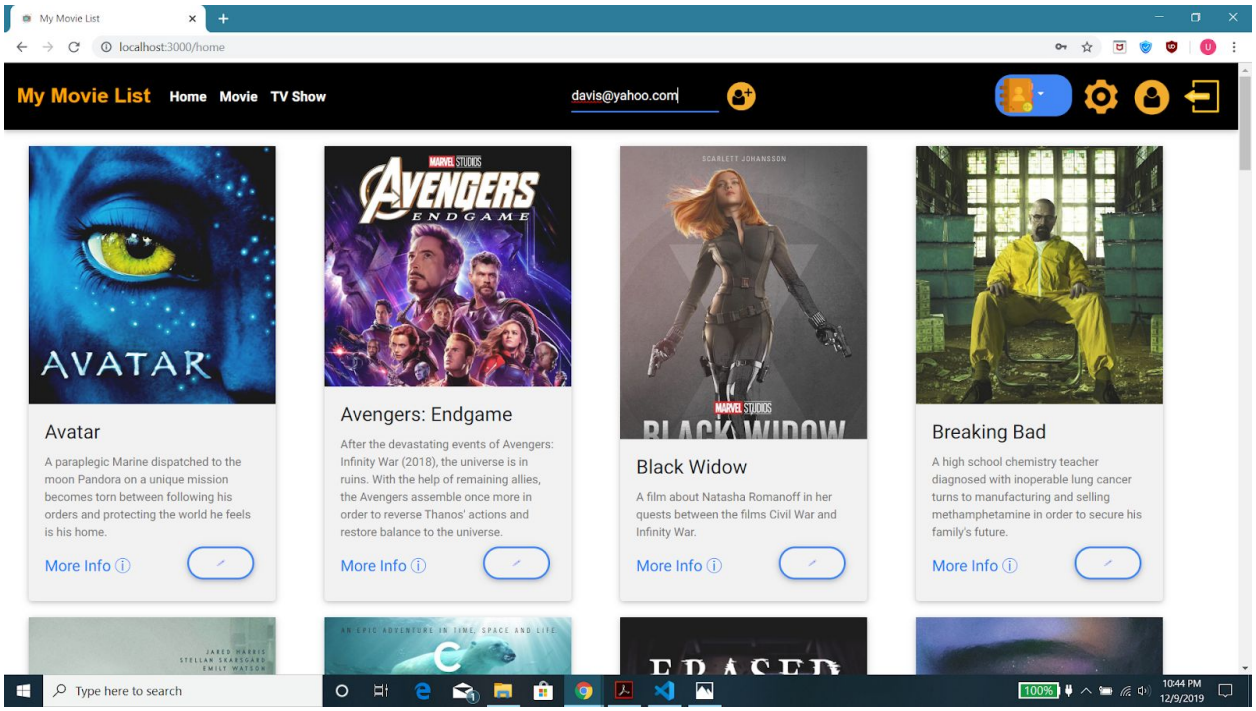
After deletion, the movie review deleted from the mysql table.



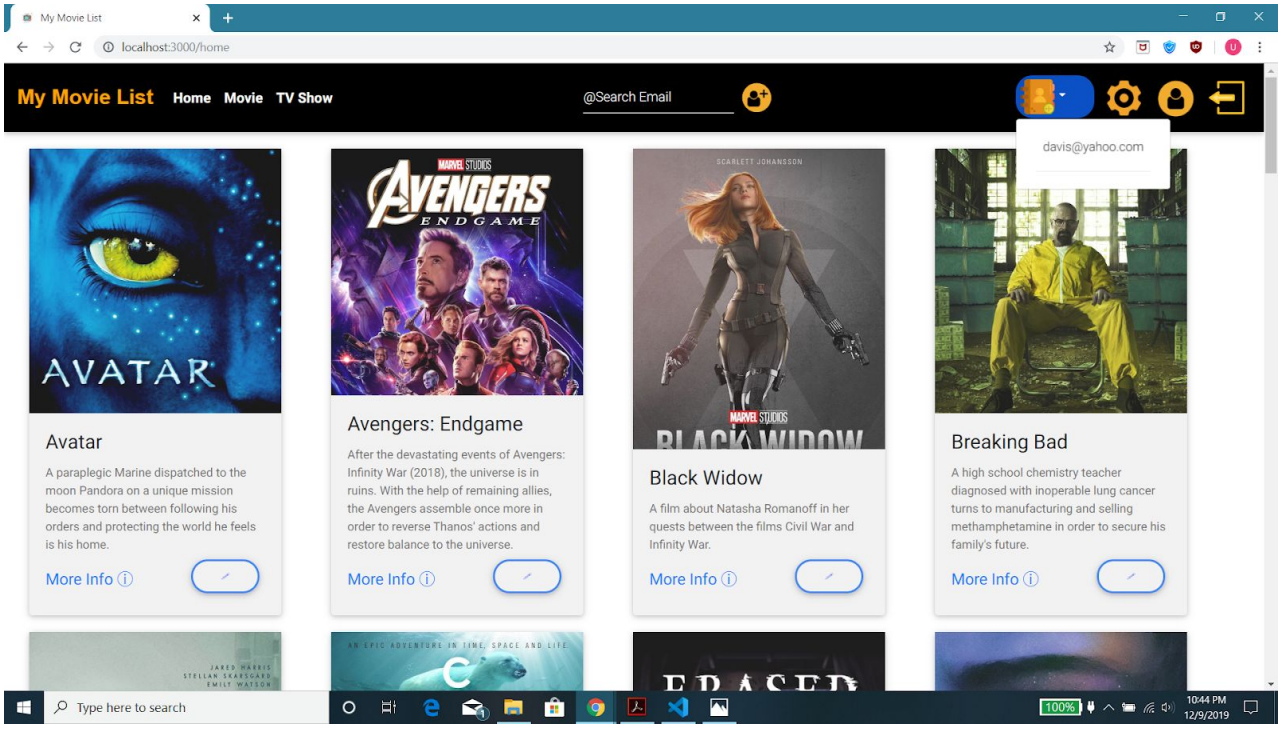
The profile page after the user “kyrie11irving@gmail.com” deletes the movie refreshes and shows that the movie review is deleted.



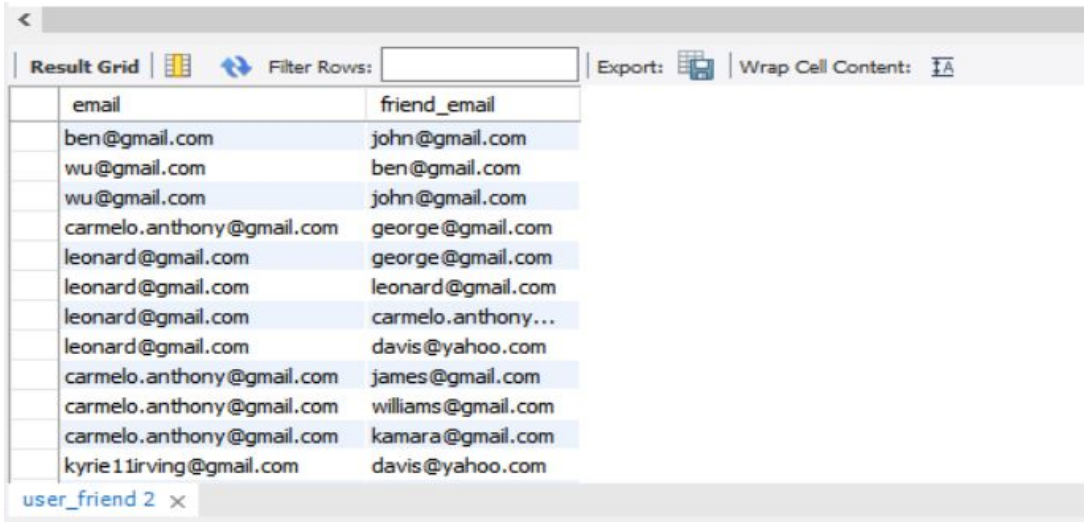
From the search bar a user can add other users as friends.



davis@yahoo.com is another user in the database, we can add the user by typing in davis@yahoo.com in the input of search bar and click on the add user button next to it to add a friend.



After adding a user, the dropdown menu of friend's list shows davis@yahoo.com.

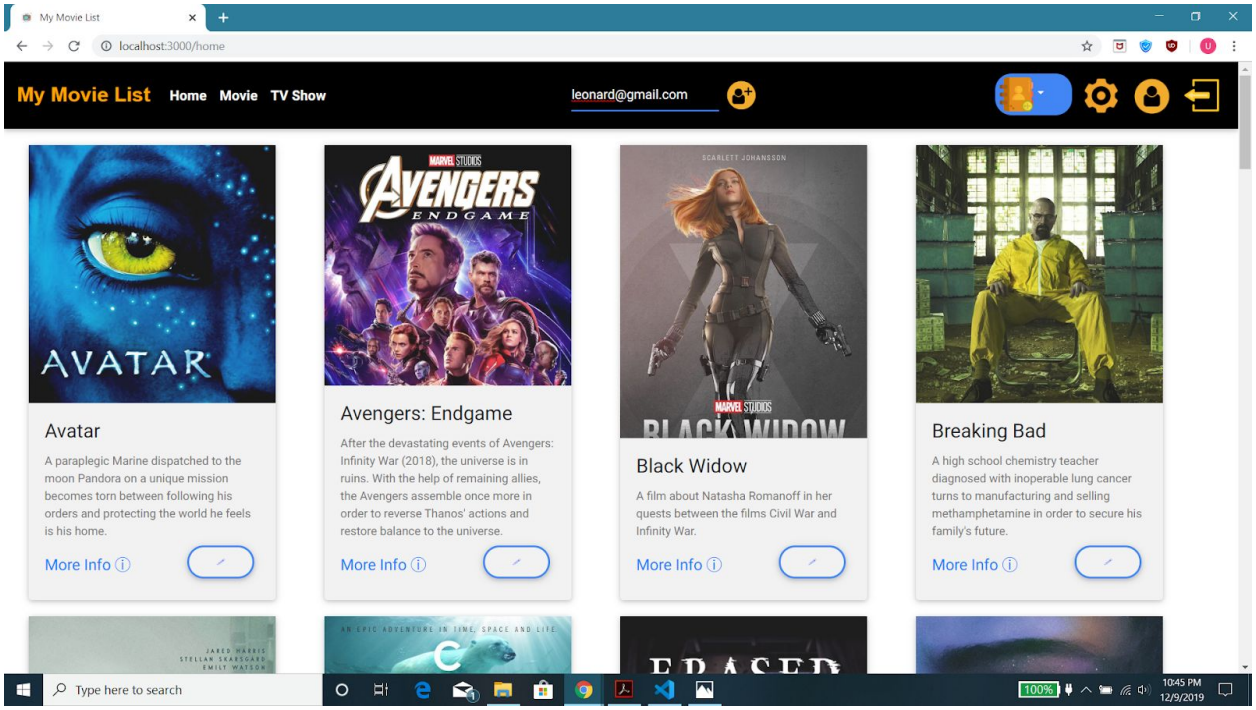


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

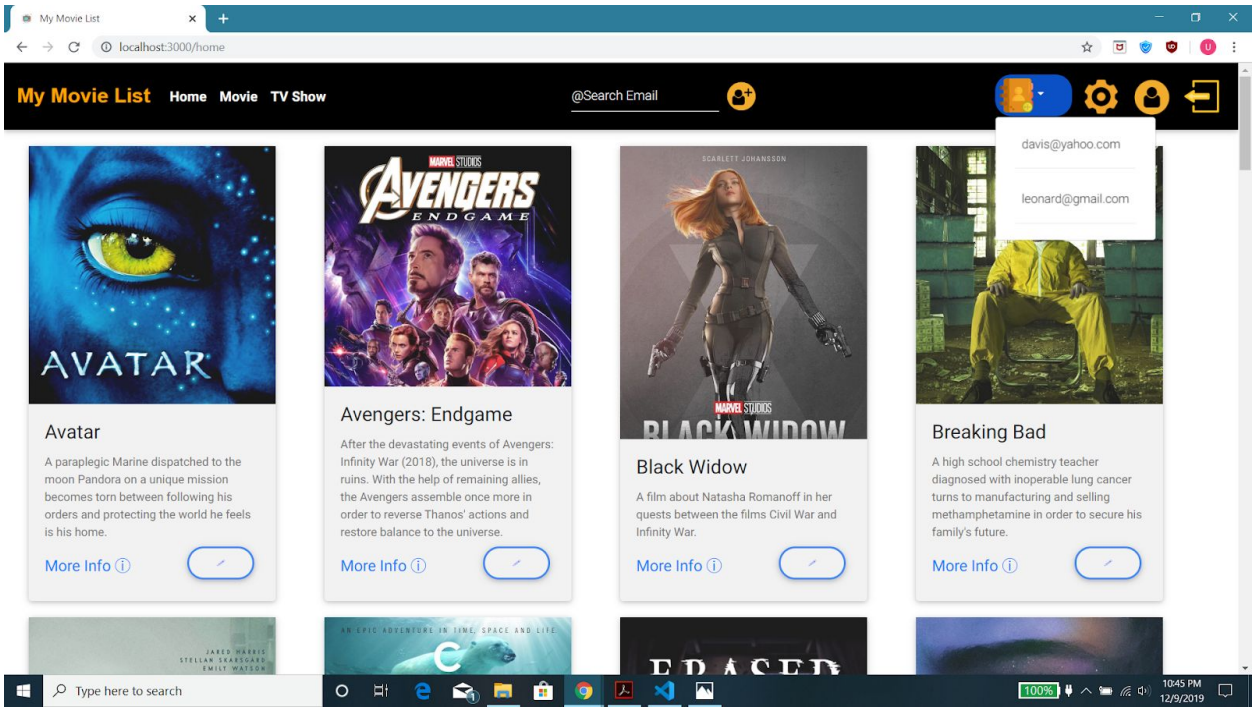
email	friend_email
ben@gmail.com	john@gmail.com
wu@gmail.com	ben@gmail.com
wu@gmail.com	john@gmail.com
carmelo.anthony@gmail.com	george@gmail.com
leonard@gmail.com	george@gmail.com
leonard@gmail.com	leonard@gmail.com
leonard@gmail.com	carmelo.anthony...
leonard@gmail.com	davis@yahoo.com
carmelo.anthony@gmail.com	james@gmail.com
carmelo.anthony@gmail.com	williams@gmail.com
carmelo.anthony@gmail.com	kamara@gmail.com
kyrie11irving@gmail.com	davis@yahoo.com

user_friend 2

As can be seen, now kyrie11irving@gmail.com has a friend davis@yahoo.com in the database.



A user can add many friends, user can add another user as a friend. As in this, a user adds leonard@gmail.com to his friend's list.

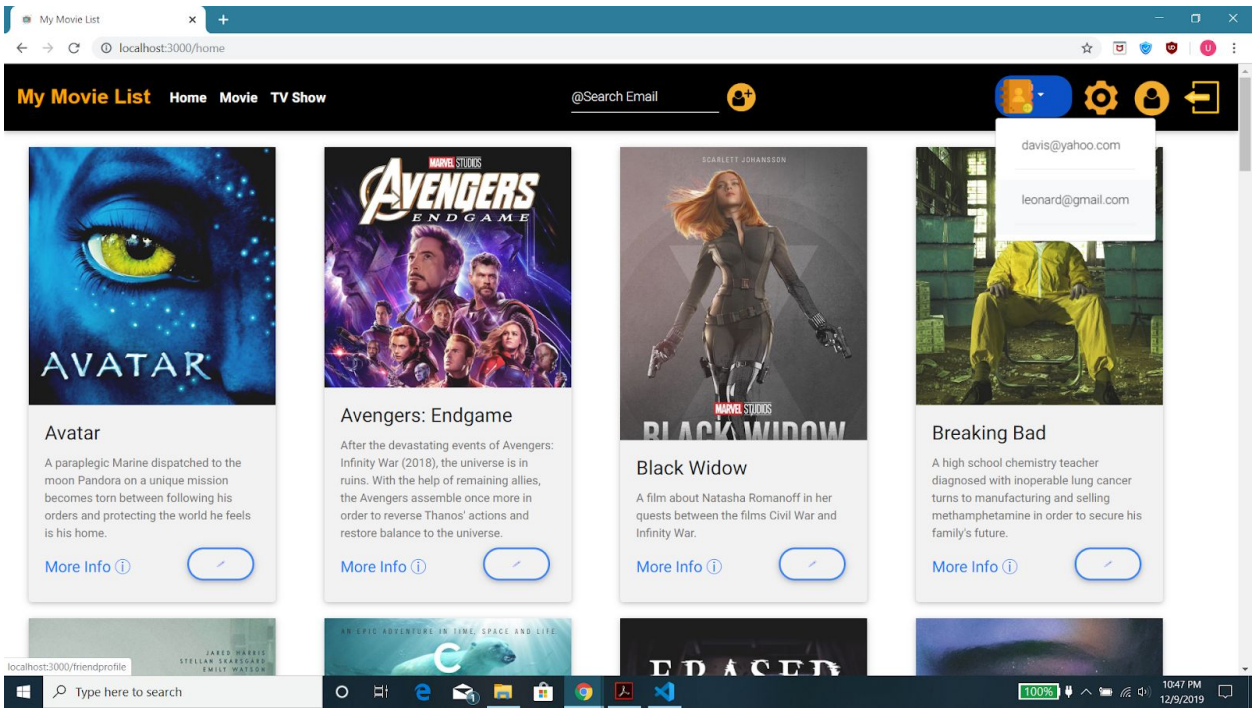


From the dropdown menu we can see the leonard@gmail.com is also added to the friend's list.

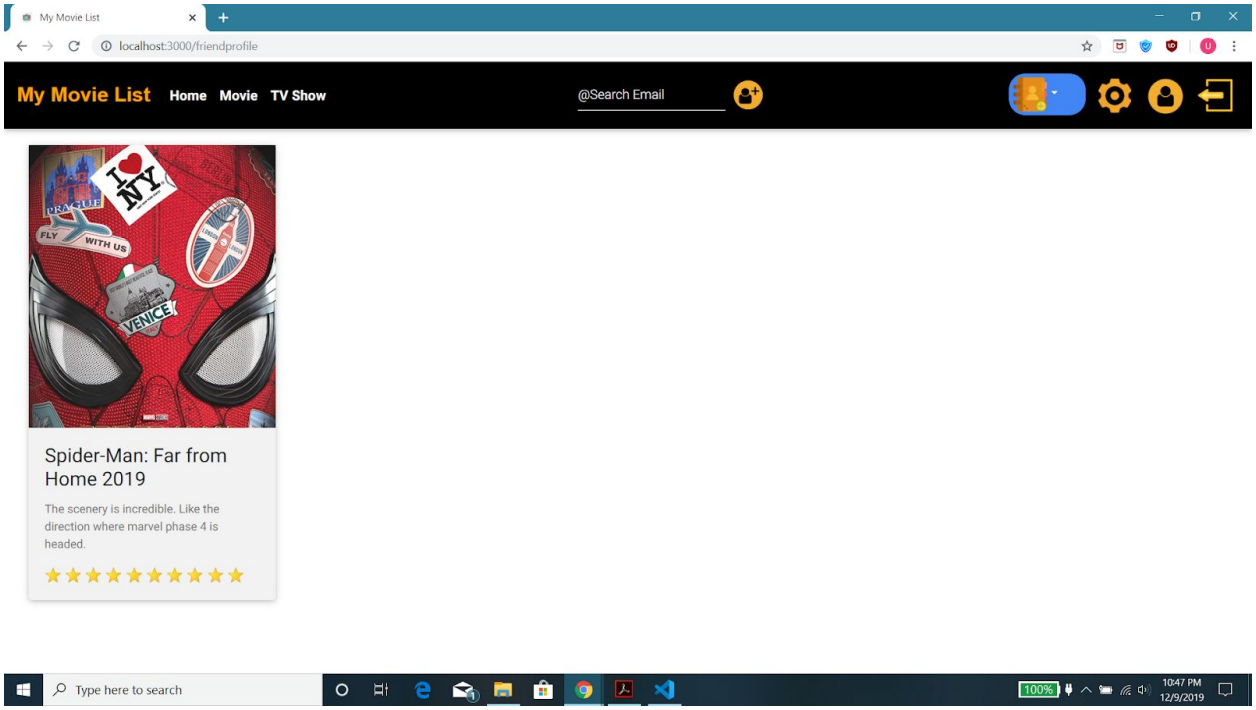
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

email	friend_email
wu@gmail.com	ben@gmail.com
wu@gmail.com	john@gmail.com
carmelo.anthony@gmail.com	george@gmail.com
leonard@gmail.com	george@gmail.com
leonard@gmail.com	leonard@gmail.com
leonard@gmail.com	carmelo.anthony...
leonard@gmail.com	davis@yahoo.com
carmelo.anthony@gmail.com	james@gmail.com
carmelo.anthony@gmail.com	williams@gmail.com
carmelo.anthony@gmail.com	kamara@gmail.com
kyrie11irving@gmail.com	davis@yahoo.com
kyrie11irving@gmail.com	leonard@gmail.com

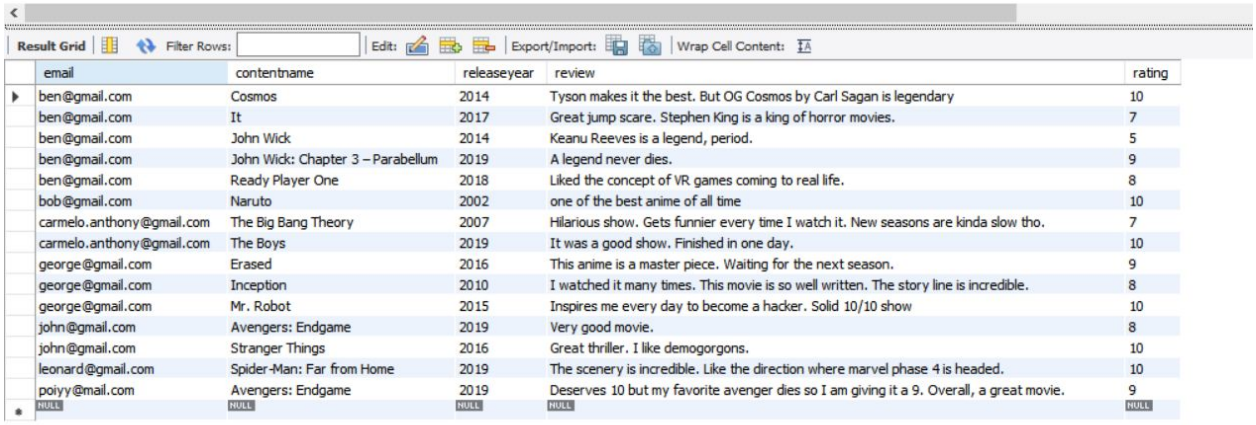
Leonard@gmail.com is added to kyrie11irving@gmail.com in MySql.



When you click on leonard@gmail.com from the dropdown menu, you can see the reviews given by leonard@gmail.com



Leonard@gmail.com has only rated Spider-Man: Far from Home which shows when a user clicks on leonard@gmail.com from the dropdown menu.



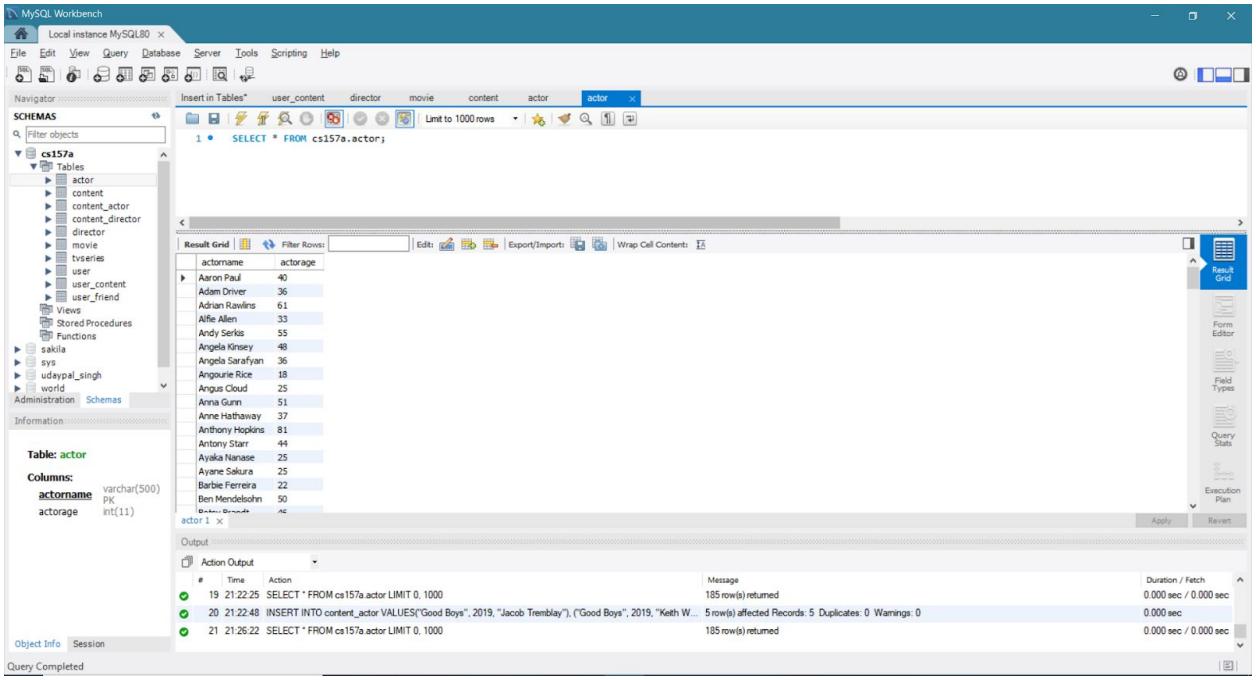
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

email	contentname	releaseyear	review	rating
ben@gmail.com	Cosmos	2014	Tyson makes it the best. But OG Cosmos by Carl Sagan is legendary	10
ben@gmail.com	It	2017	Great jump scare. Stephen King is a king of horror movies.	7
ben@gmail.com	John Wick	2014	Keanu Reeves is a legend, period.	5
ben@gmail.com	John Wick: Chapter 3 – Parabellum	2019	A legend never dies.	9
ben@gmail.com	Ready Player One	2018	Liked the concept of VR games coming to real life.	8
bob@gmail.com	Naruto	2002	one of the best anime of all time	10
carmelo.anthony@gmail.com	The Big Bang Theory	2007	Hilarious show. Gets funnier every time I watch it. New seasons are kinda slow tho.	7
carmelo.anthony@gmail.com	The Boys	2019	It was a good show. Finished in one day.	10
george@gmail.com	Erased	2016	This anime is a master piece. Waiting for the next season.	9
george@gmail.com	Inception	2010	I watched it many times. This movie is so well written. The story line is incredible.	8
george@gmail.com	Mr. Robot	2015	Inspires me every day to become a hacker. Solid 10/10 show	10
john@gmail.com	Avengers: Endgame	2019	Very good movie.	8
john@gmail.com	Stranger Things	2016	Great thriller. I like demogorgons.	10
leonard@gmail.com	Spider-Man: Far from Home	2019	The scenery is incredible. Like the direction where marvel phase 4 is headed.	10
poiyy@mail.com	Avengers: Endgame	2019	Deserves 10 but my favorite avenger dies so I am giving it a 9. Overall, a great movie.	9
NULL	NULL	NULL	NULL	NULL

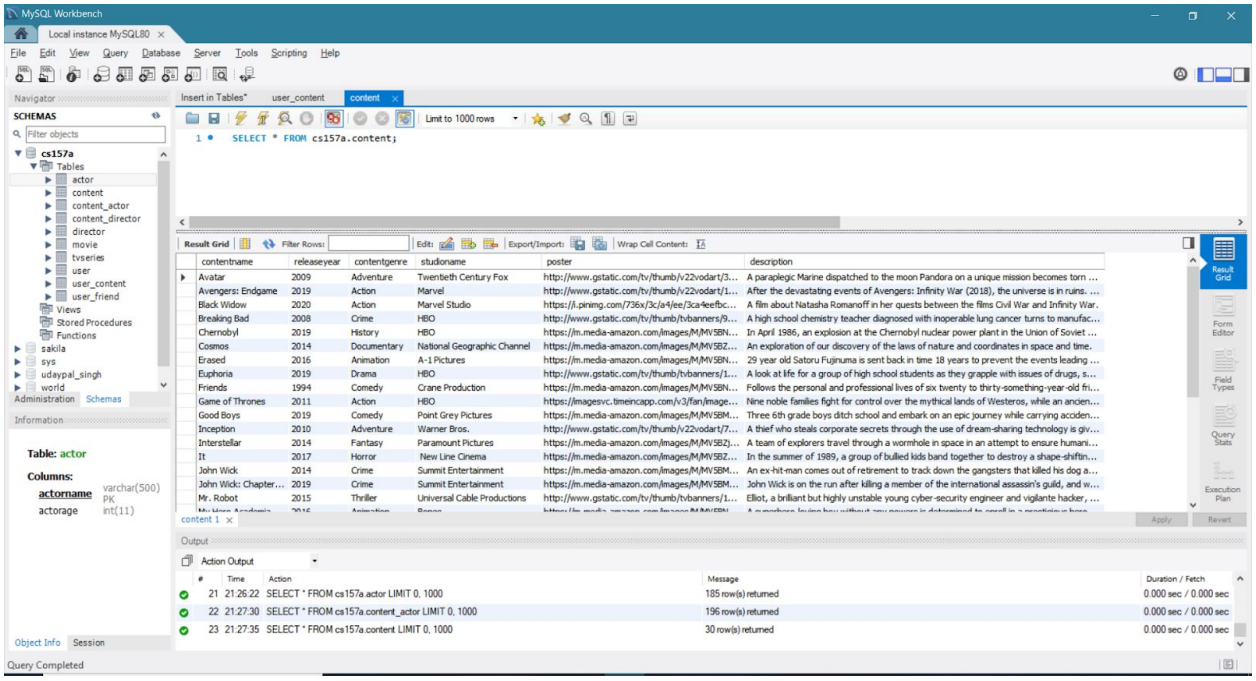
As can be seen in the screenshot of MySql, leonard@gmail.com has a review for “Spider-Man: Far from Home” and it is shown to kyrie11irving@gmail.com from friend’s list.

Screenshot of MySQL statement:

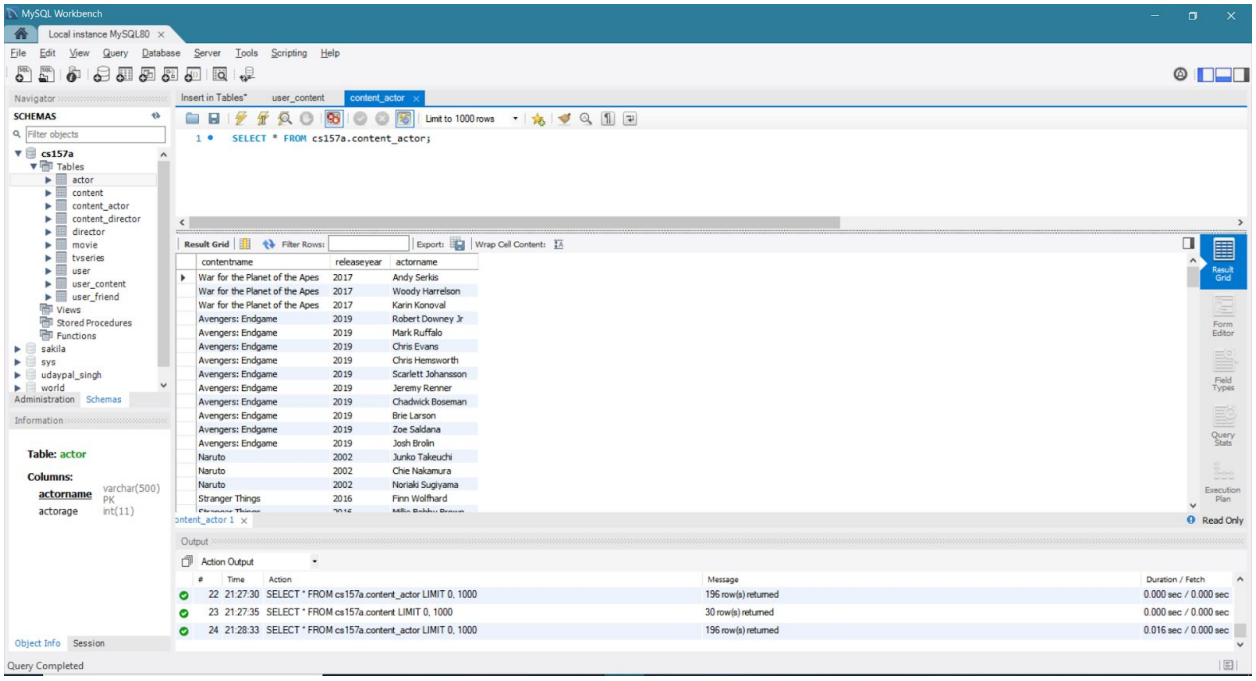
Actor



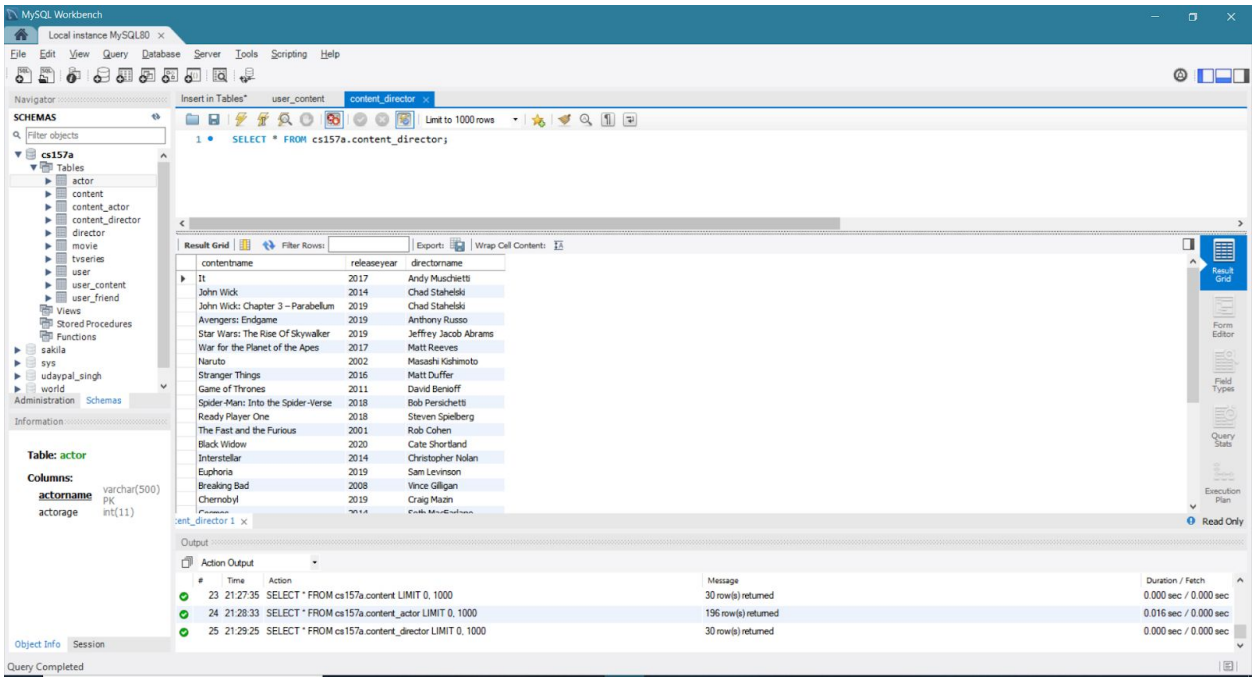
content



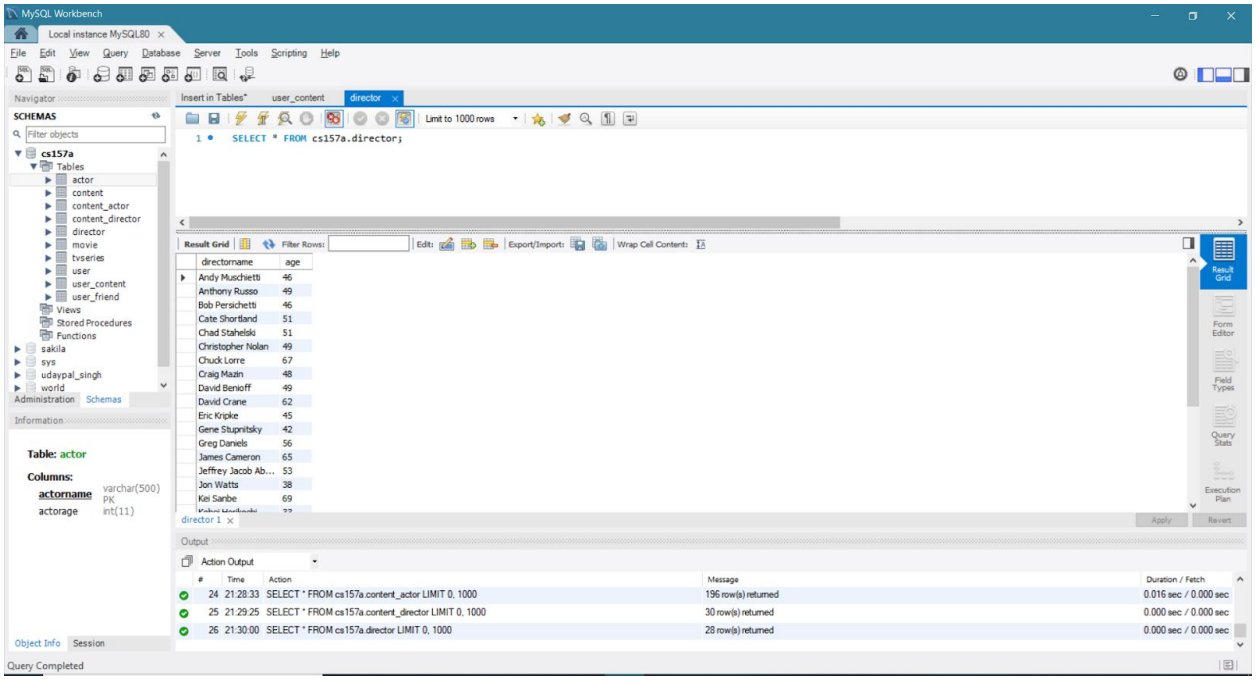
content_actor



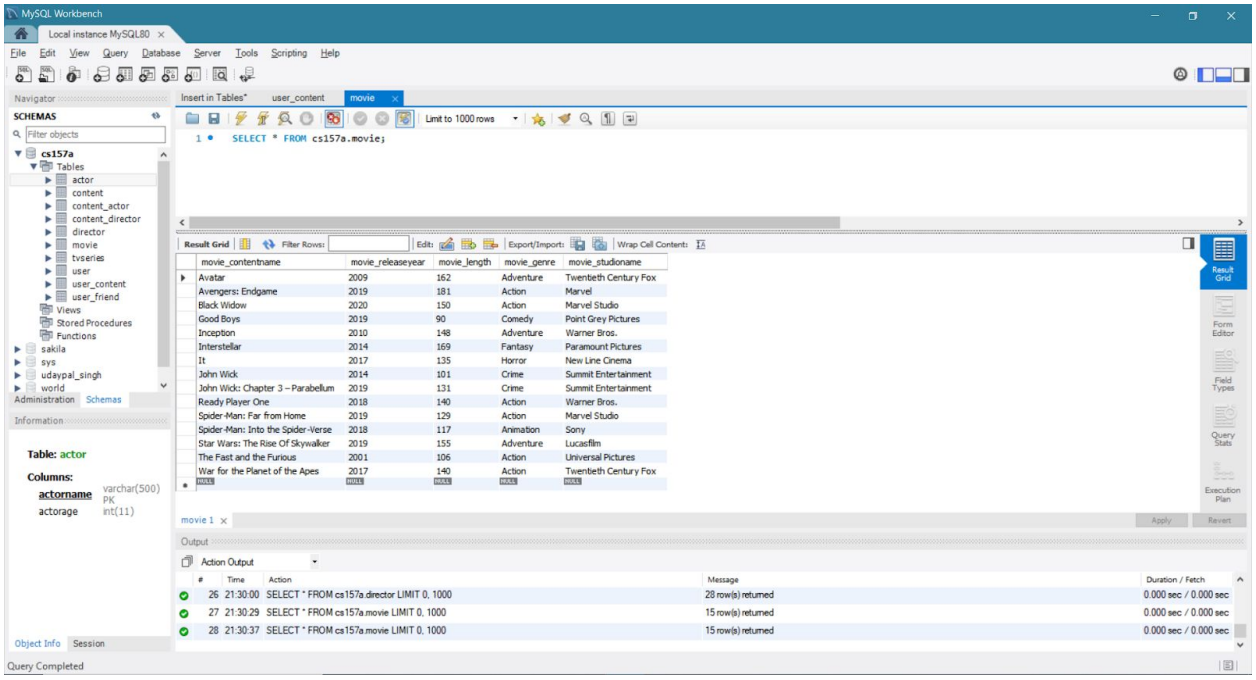
content_director



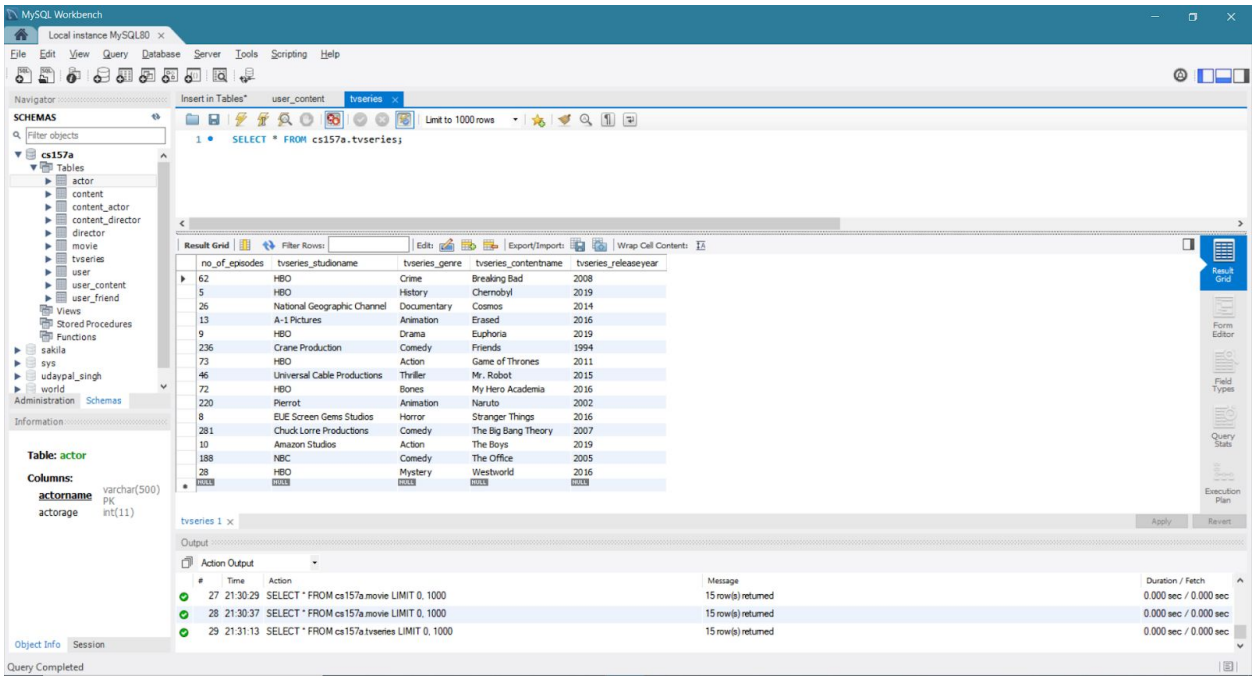
Director



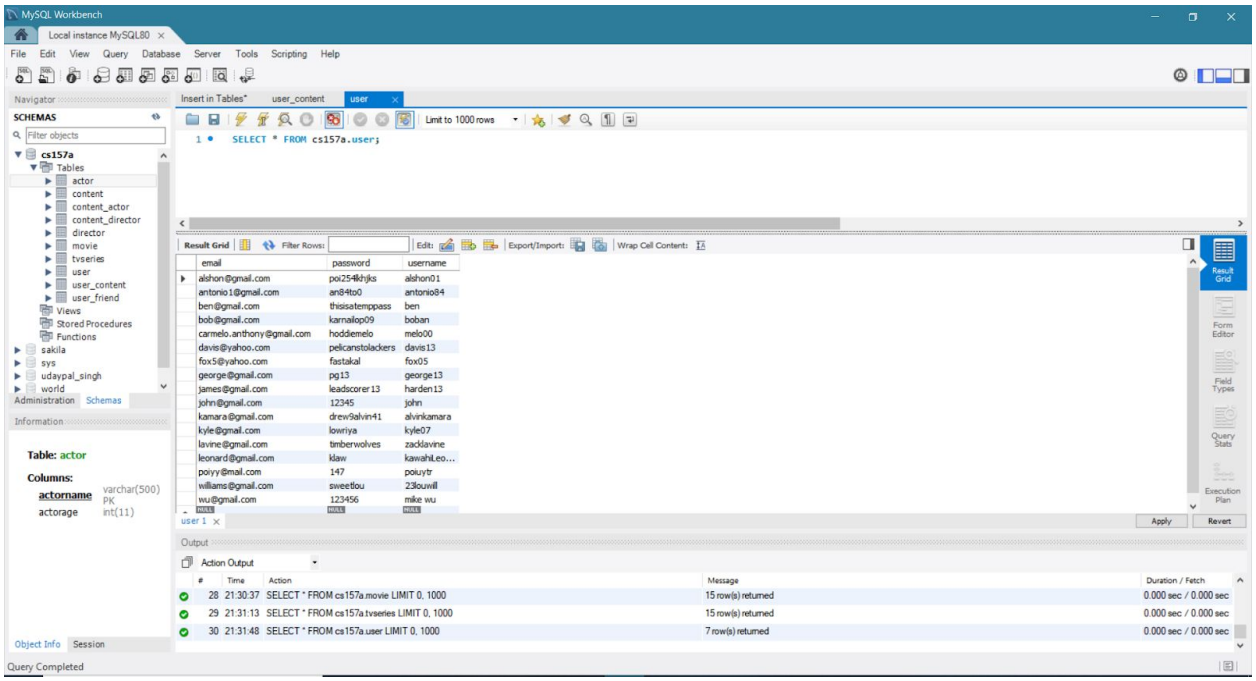
movie



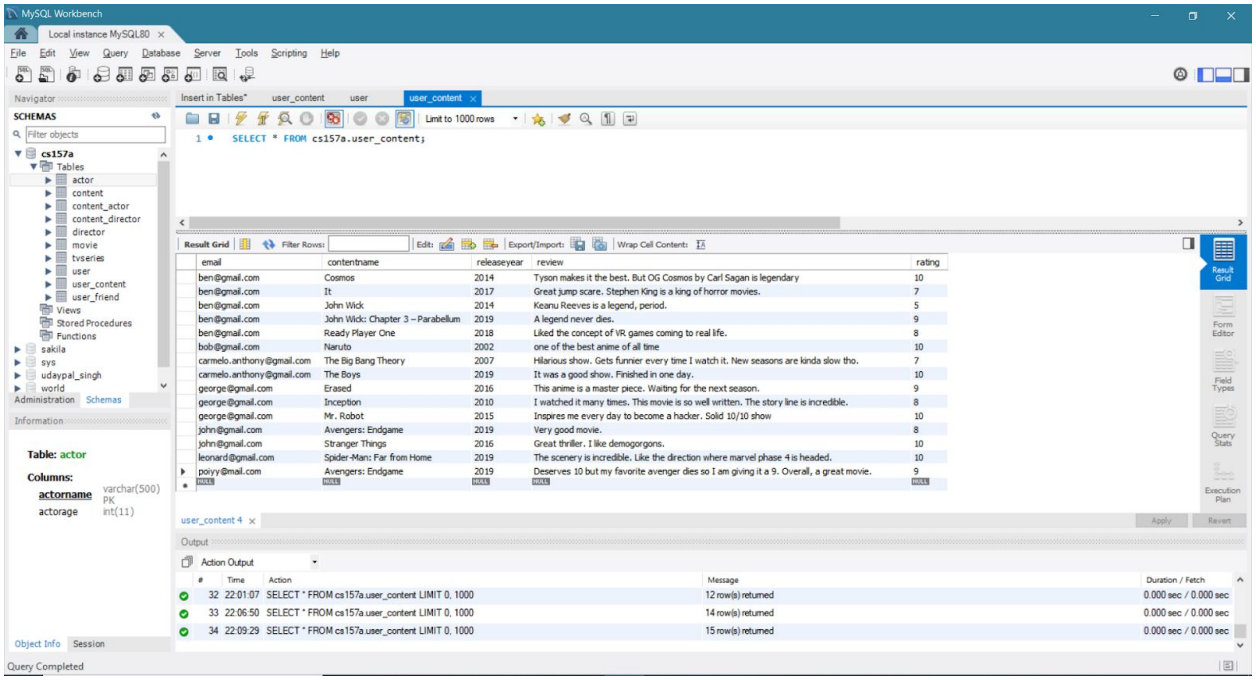
Tvseries



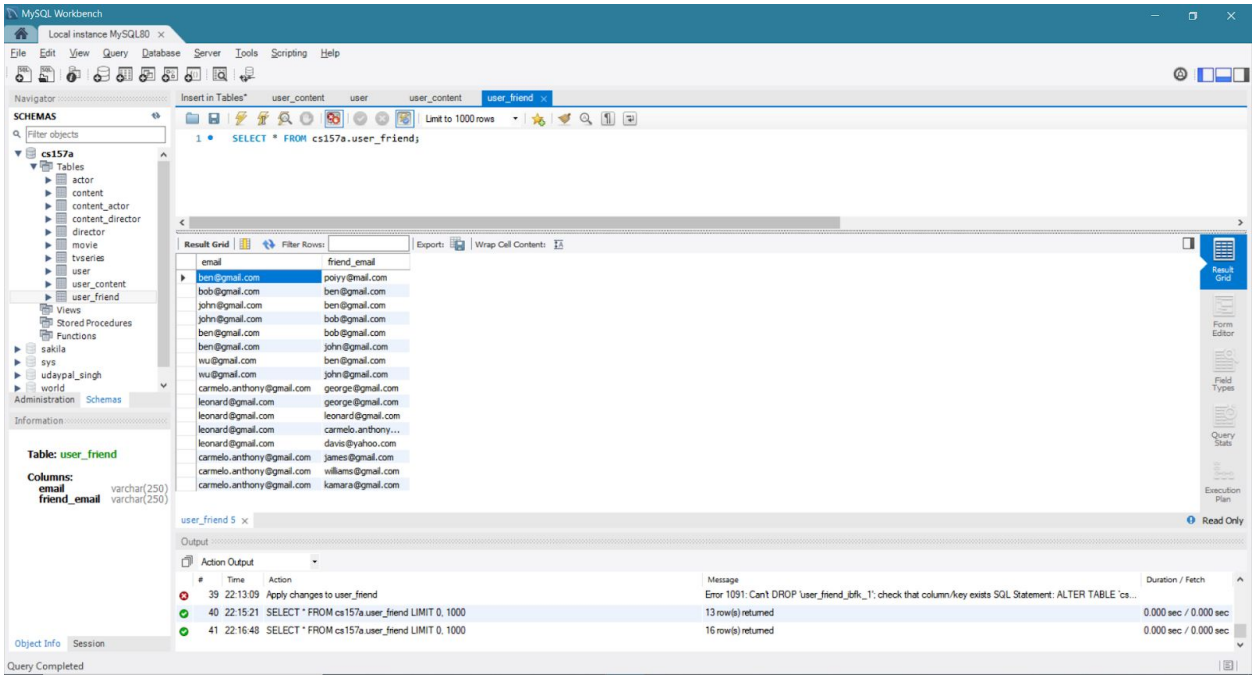
User



User_content


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SCHEMAS
cs157a
Tables
actor
content
content_actor
content_director
director
movie
tvseries
user
user_content
user_friend
Views
Stored Procedures
Functions
sakila
sys
udaypal_singh
world
Table: actor
Columns:
actorname varchar(500) PK
actorage int(11)
Insert in Tables* user_content user user_content
Limit to 1000 rows
1 • SELECT * FROM cs157a.user_content;
Result Grid
email contentname releaseyear review rating
ben@gmail.com Cosmos 2014 Tyson makes it the best. But OG Cosmos by Carl Sagan is legendary 10
ben@gmail.com It 2017 Great jump scare. Stephen King is a king of horror movies. 7
ben@gmail.com John Wick 2014 Keanu Reeves is a legend, period. 5
ben@gmail.com John Wick: Chapter 3 – Parabellum 2019 A legend never dies. 9
ben@gmail.com Ready Player One 2018 Liked the concept of VR games coming to real life. 8
bob@gmail.com Naruto 2002 one of the best anime of all time 10
carmelo.anthony@gmail.com The Big Bang Theory 2007 Hilarious show. Gets funnier every time I watch it. New seasons are kinda slow tho. 7
carmelo.anthony@gmail.com The Boys 2019 It was a good show. Finished in one day. 10
george@gmail.com Erased 2016 This anime is a master piece. Waiting for the next season. 9
george@gmail.com Inception 2010 I watched it many times. This movie is so well written. The story line is incredible. 8
george@gmail.com Mr. Robot 2015 Inspires me every day to become a hacker. Solid 10/10 show 10
john@gmail.com Avengers: Endgame 2019 Very good movie. 8
john@gmail.com Stranger Things 2016 Great thriller. I like demogorgons. 10
leonard@gmail.com Spider-Man: Far from Home 2019 The scenery is incredible. Like the direction where marvel phase 4 is headed. 10
poiyy@mail.com Avengers: Endgame 2019 Deserves 10 but my favorite avenger dies so I am giving it a 9. Overall, a great movie. 9
user_content 4
Output
Action Output
Time Action Message Duration / Fetch
32 22:01:07 SELECT * FROM cs157a.user_content LIMIT 0, 1000 12 row(s) returned 0.000 sec / 0.000 sec
33 22:06:50 SELECT * FROM cs157a.user_content LIMIT 0, 1000 14 row(s) returned 0.000 sec / 0.000 sec
34 22:09:29 SELECT * FROM cs157a.user_content LIMIT 0, 1000 15 row(s) returned 0.000 sec / 0.000 sec
Query Completed

User_friend


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SCHEMAS
cs157a
Tables
actor
content
content_actor
content_director
director
movie
tvseries
user
user_content
user_friend
Views
Stored Procedures
Functions
sakila
sys
udaypal_singh
world
Table: user_friend
Columns:
email varchar(250)
friend_email varchar(250)
Insert in Tables* user_content user user_content user_friend
Limit to 1000 rows
1 • SELECT * FROM cs157a.user_friend;
Result Grid
email friend_email
ben@gmail.com poiyy@mail.com
bob@gmail.com ben@gmail.com
john@gmail.com ben@gmail.com
john@gmail.com bob@gmail.com
ben@gmail.com bob@gmail.com
ben@gmail.com john@gmail.com
wu@gmail.com ben@gmail.com
wu@gmail.com john@gmail.com
carmelo.anthony@gmail.com george@gmail.com
leonard@gmail.com george@gmail.com
leonard@gmail.com leonard@gmail.com
leonard@gmail.com carmelo.anthony...
leonard@gmail.com davis@yahoo.com
carmelo.anthony@gmail.com james@gmail.com
carmelo.anthony@gmail.com williams@gmail.com
carmelo.anthony@gmail.com kamara@gmail.com
user_friend 5
Read Only
Output
Action Output
Time Action Message Duration / Fetch
39 22:13:09 Apply changes to user_friend Error 1091: Can't DROP 'user_friend_ibfk_1'; check that column/key exists SQL Statement: ALTER TABLE 'cs...
40 22:15:21 SELECT * FROM cs157a.user_friend LIMIT 0, 1000 13 row(s) returned 0.000 sec / 0.000 sec
41 22:16:48 SELECT * FROM cs157a.user_friend LIMIT 0, 1000 16 row(s) returned 0.000 sec / 0.000 sec
Query Completed

Normalization: BCNF

Actor

Actor(actorname, actorage)
Let A = actorname, B = actorage
Functional Dependency: A->B
A->B is a nontrivial functional dependency and A is a superkey; therefore the Actor table is in BCNF.

content

content(contentname, releaseyear, conentgenre, studioname, poster, description)
Let A = contentname, B = releaseyear, C = contentgenre, D = studioname, E = poster, F = description
Functional Dependency: A->BCDEF
A->BCDEF is a nontrivial functional dependency and A is a superkey; therefore the content table is in BCNF.

content_actor

content_actor(contentname, releaseyear, actorname)
Let A = contentname, B = releaseyear, C = actorname
Functional Dependency: AC->B
AC->B is a nontrivial functional dependency and AC is a superkey; therefore the content_actor table is in BCNF.

content_director

content_director(contentname, releaseyear, directorname)
Let A = contentname, B = releaseyear, C = directorname
Functional Dependency: A->BC
A->BC is a nontrivial functional dependency and A is a superkey; therefore the Content_director table is in BCNF.

Director

Director(directorname, age)
Let A = directorame, B = age
Functional Dependency: A->B
A->B is a nontrivial functional dependency and A is a superkey; therefore the Director table is in BCNF.

movie

movie(movie_contentname, movie_releaseyear, movie_length, movie_genre, movie_studioname)
Let A = movie_contentname, B = movie_releaseyear, C = movie_length,
D = movie_genre, E = movie_studioname
Functional Dependency: A->BCDEF
A->BCDEF is a nontrivial functional dependency and A is a superkey; therefore the movie table is in BCNF.

Tvseries

Tvseries(no_of_episodes, tvseries_studioname, tvseries_genre, tvseries_contentname tvseries_releaseyear)
Let A = no_of_episodes, B = tvseries_studioname, C = tvseries_genre,
D = tvseries_contentname, E = tvseries_releaseyear
Functional Dependency: D->ABCEF
D->ABCEF is a nontrivial functional dependency and D is a superkey; therefore the Tvseries table is in BCNF.

User

User(email, password, username)
Let A = email, B = password, C = username
Functional Dependency: A -> BC, C ->AB
Both functional dependencies are nontrivial and A and C are superkeys; therefore the User table is in BCNF.

User_content

User_content(email, contentname, releaseyear, review, rating)
Let A = email, B = contentname, C = releaseyear, D = review, E = rating
Functional Dependency = A->BCDE
A->BCDE is a nontrivial functional dependency and D is a superkey; therefore the User_content table is in BCNF.

User_friend

User_friend(email, friend_email)
Let A = email, B = friend_email
Functional Dependencies = A ->B and B->A
A->B and B->A are nontrivial functional dependencies and both are superkeys; therefore the User_friend table is in BCNF.

Conclusion:

Jiawei Zhang(Team leader):
Throughout this database project, it was my first time completing a full stack website application. It was quite a challenging project because I have zero experience in terms of integrating front end and backend with MySQL database. I was responsible for front end development and backend connection. The hardest part for me was to fetch the data from MySQL database, parse to JSON format and then display the result data with style in CSS. However as I finished the task I gained a better understanding of Three-Tier architecture and Restful API design. Besides the technical aspect, it was also my first time being a team leader. I was able to bring my team to deliver the project in time and improve my management skills as well.

Udaypal Singh:
The development process of this website, with MySql database and React, has taught me many valuable skills which I can carry on with me. Constructing an ER diagram is a base for the

project, which I believe did a really good job. Starting the project from scratch and connecting backend from Sql to frontend using node.js and express was not easily achieved as I had no previous experience with this. I worked on retrieving the data from the backend including list of content, movies, tvseries, and using union of the table content UNION movie, and content UNION tvseries and then using JOIN to querying actors and directors from these complex queries were a hassle. I was also responsible for inserting, editing, and deleting the review and rating for a user reviewed movie from the frontend to backend. I also faced difficulties when working with the UI; for example displaying the content cards in-line, and designing moreinfo, rating on top of the background image involved many UI design complexities. Despite the challenges, I believe I learned a lot, and now I feel confident that I can develop a website with a solid database structure.

Adan Hernandez:
Throughout the course of the project I gained very valuable skills like SQL programming and React. It was very challenging, and being the first time I worked on a three tier architecture project from scratch I had a hard time learning the skills needed to complete the project. The making of the ER diagram and the documentation of everything were other key skills that I learned that I know will benefit me in the long run. In other classes, there usually isn't documentation that goes into it, however, it is an important skill because it is expected to document everything in internships. I worked on some of the front-end, and after this project I know that I really enjoy front-end. I especially like being able to connect the front-end with back-end and see how it all comes together.

Future Improvement of the Application:

There are some improvements that can be made within the application. For example, the database of our application is based on local storage MySQL database. If the user who runs the application does not have the data set up in MySQL, then the application will have blank information when the user enters on the homepage of our application. In the future we could set up a cloud database and share the resource via cloud storage, hence every user who enter our application will have the same content showed on homepage. In the future we could also have movie database API setup in our application. This would allow our application to scale and manipulate a lot more data overtime.