

PROJECT REPORT
ON
ONLINE MOVIE RATING SYSTEM
(M-REVIEWS)

B.Tech (CE) SEM – V
In the subject of
ADVANCED TECHNOLOGIES

By
Ronak Padaliya (CE094) (19CEUEG041)
Darshan Parmar (CE099) (19CEUOS050)
Dhyey Patel (CE109) (19CEUOS145)

Under the Guidance of
Prof. Prashant M. Jadav
Prof. Sidhharth P. Shah



Department of Computer Engineering
Faculty of Technology,
Dharmsinh Desai University, Nadiad.



DHARMSINH DESAI UNIVERSITY

College Road, NADIAD-387001(Gujarat)

CERTIFICATE

This is to certify that the term work carried out in the subject of

Advanced Technologies and recorded in this report is

Bonafide work of

Mr. Ronak Padaliya (Roll No.: CE094, ID: 19CEUEG041)

Mr. Darshan Parmar (Roll No.: CE099, ID: 19CEUOS050)

And

Mr. Dhyey Patel (Roll No.: CE109, ID: 19CEUOS145)

of B.Tech Semester V

in the branch of Computer Engineering during the

academic year 2021-22.

Prof. Prashant M Jadav

Project Guide and Associate Professor
Faculty of Technology,
Dharmsinh Desai University,
Nadiad.

Dr. C. K. Bhensdadia

Head of the CE Dept.
Faculty of Technology,
Dharmsinh Desai University,
Nadiad.

Index

Certificate.....	2.
Abstract.....	4.
Introduction.....	5.
Software Requirements Specifications (SRS).....	6.
Design	
Dataflow Diagram.....	7.
ER Diagram.....	9.
Data Dictionary.....	10.
Implementation Details	
Modules.....	11.
Function prototypes.....	12.
Algorithm and flowchart.....	17.
Testing.....	18.
Screenshots.....	19.
Conclusion.....	25.
Limitations and Future Extensions.....	26.
Bibliography.....	27.

1. **Abstract:**

- M-Reviews is an online movie rating and information collection system build to collect data about public interest, from their reviews about movies and Web-Shows. Here user can give rating and review about movies by simply creating an account or simply just login with existing account.

2. Introduction:

- **Brief Introduction:**

- M-Reviews is a system in which user can explore any movies or tv-shows also can give rating and write reviews about it.

- **Technologies:**

- react.js
- node.js
- MongoDB
- express.js
- bootstrap
- material-ui
- CSS
- MongoDB Atlass

- **Tools:**

- Visual Studio Code
- Git
- MongoDB Compass

- **Platform:**

- GitHub
- Local Development Server

3. Software Requirements Specification(SRS):

1. Manage User:

1.1 Sign-up:

Input : Enter details for Sign-up

Output : Account Created

1.2 Login:

Input : Enter user email and password

Output : Login successful if valid credentials else display error

2. Movie Search and show details:

1.1 Movie search:

Input : Title of the movie

Output : list of movies with similar name or title with user requirement they can sort the list accordingly.

1.2 Show Details:

Input : User selection

Output : show details about selected movie or TV-show

3. Rating Management:

1.1 Give rating and reviews:

Description: User can rate a particular movie between 1 to 10 and write reviews about the movie, this rating is stored in database and with average of every single review the final rating of the movie is shown.

Input : users' review

Output : store it and update to database

1.2 Update user rating:

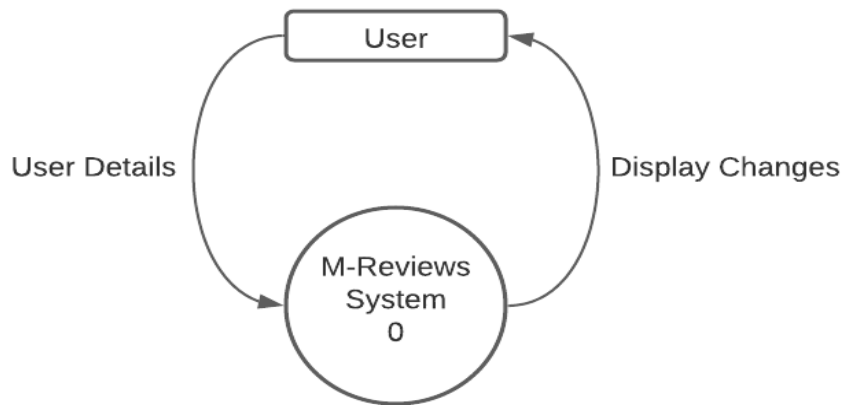
Input : change their ratings with newly given rating

Output : update ratings to database

4. Design:

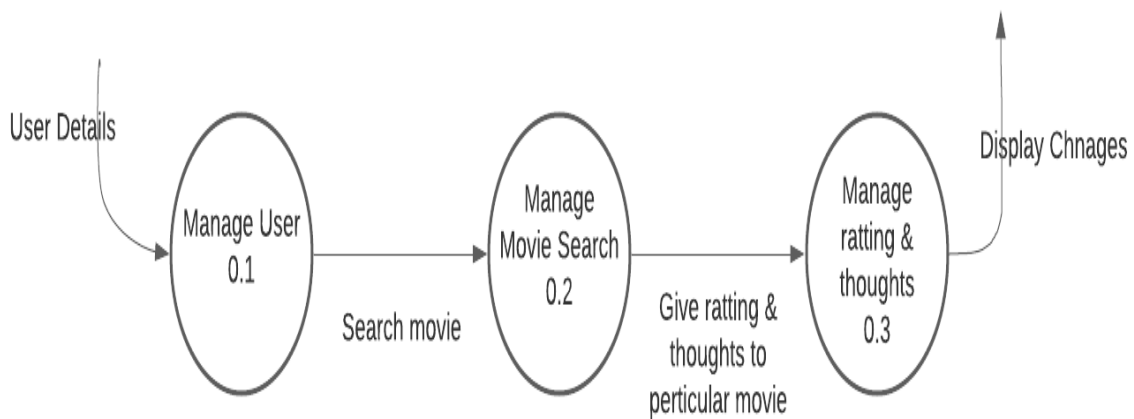
i. Data-flow Diagram:

➤ Level 0:



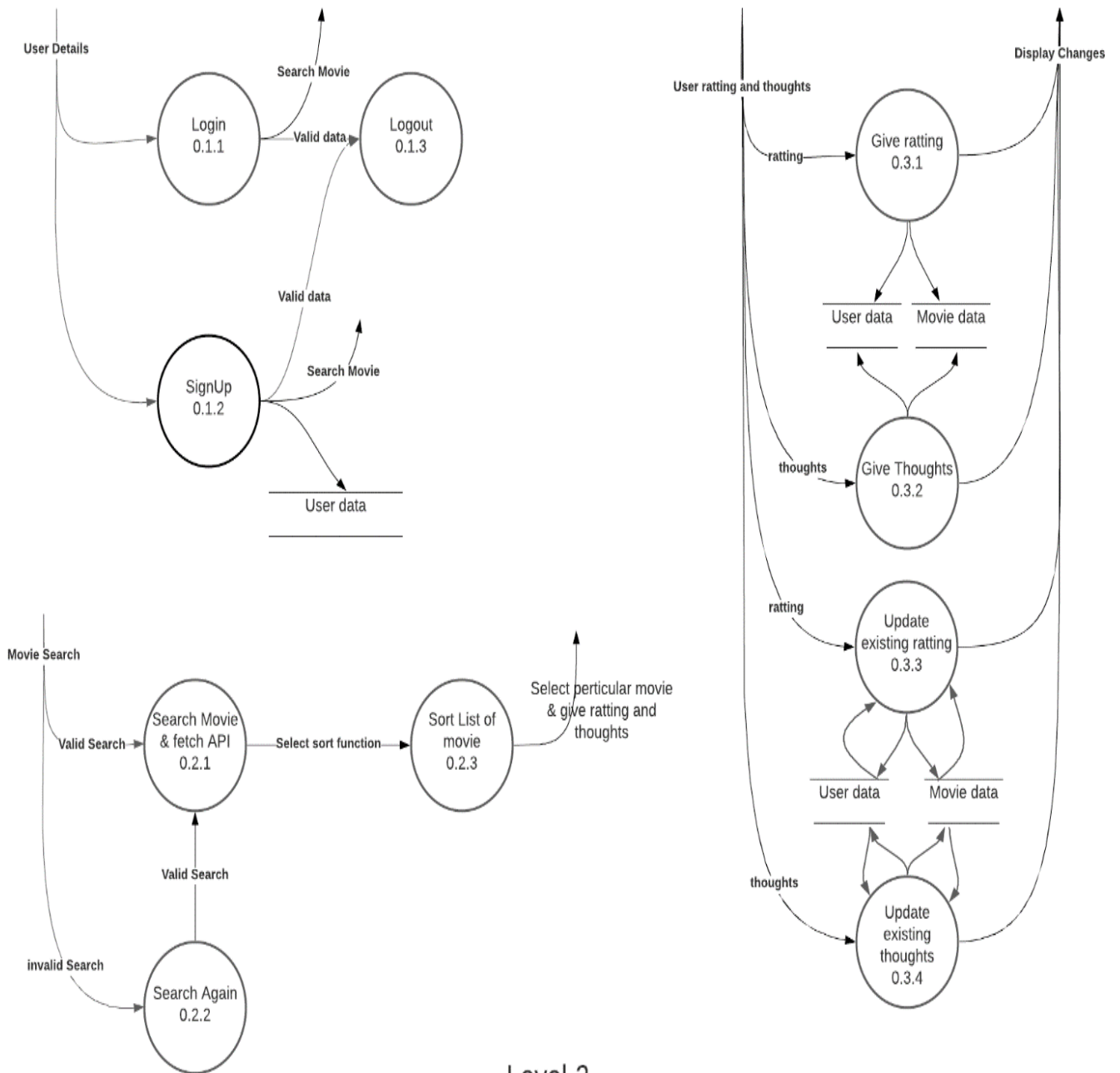
Level-0

➤ Level 1:



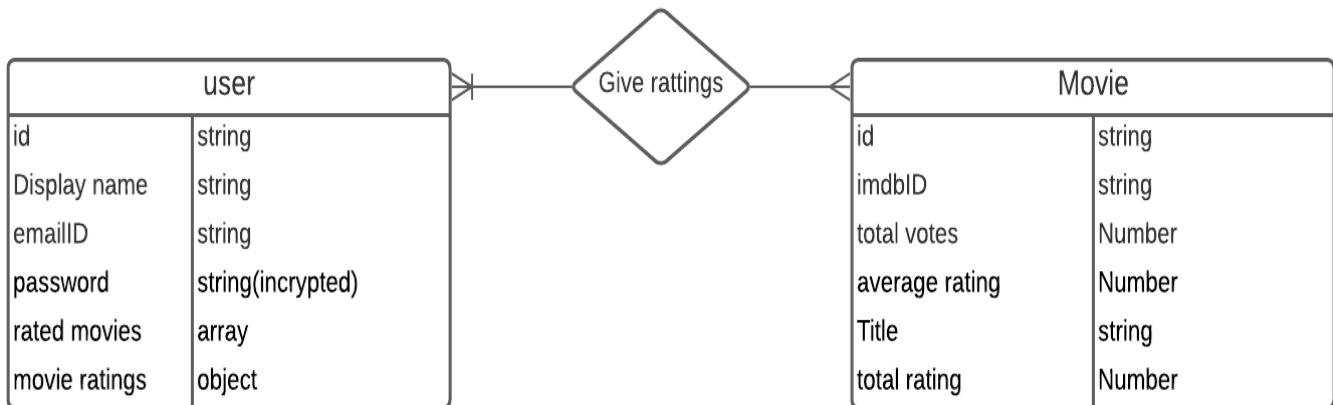
Level-1

➤ Level 2:



Level-2

ii. ER Diagram:



iii. Data Dictionary:

➤ User-Scheme:

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

// Here nosql mongodb database is used.
// User is document to store the data of all client who signedUp using Email.
const User = new Schema(
  {
    displayName: { type: String, required: true},
    email: { type: String, required: true},
    password: { type: String, required: true},
    ratedmovie: {type: Array, default:[]},
    userratting: {type: Array, default:[]},
    // It should contain object id of movie-model table, user's ratting.
  },
  {
    timestamps: true
  }
)

module.exports = mongoose.model('usersList', User)
```

➤ Movie-model:

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

const Movie = new Schema(
  {
    title: { type: String, required: true },// title of movie
    avgratting: { type: Number, required: true },// average rating of movie
    imdbid: { type: String, required: true },// imdbid of movie from imdb.com
    totalratesum: { type: Number, required: true, default:0 },// total sum of rating from all user.
    ratecount: {type: Number, required: true, default:0},// count of rate
  },
  { timestamps: true },
)

module.exports = mongoose.model('moviedata', Movie)
```

5. Implementation Details:

i. Modules:

- The system mainly consists of 2 modules
 1. User-module.
 2. Movie-module.
- These two modules are used to store all the related data of website users, so that it can be retrieve whenever needed. MongoDB(NoSQL) is used as database, node.js is used to handle database at backend and react.js is used to retrieve the data so that users can view all the data.
- **User-module**

This module helps to store data of user who signed-up to web site. This includes DisplayName, email, password and information of movies which user rates. It also helps in authentication(login/signup) of users in website.
- **Movie-module**

This module stores the information related to all movies which were rated by one or more than one users. Data includes title of movie, average rating of movie, imdbId, total sum of rating from all user and count of rates. It helps in retrieving the data of movies smoothly. Other information related to movie is fetched using key like imdbId by Restful API.

ii. Function prototypes:

➤ User-controller:

```
const User = require("../models/user-models");
// bcrypt is for encrypting passwords so make sure to download in server side dependencies.
const bcrypt = require("bcrypt");

// For signUp : createUser, verifyEmail.
// For login : getUser.

// User is created using MongoDB database when client tries to signUp.
createUser = async (req, res) => {
  const body = req.body;
  if (!body) {
    return res.status(400).json({
      success: false,
      error: "You must provide a User",
    });
  }
  const user = new User(body);
  if (!user) {
    return res.status(400).json({ success: false, error: err });
  }
  const salt = await bcrypt.genSalt(8);
  user.password = await bcrypt.hash(user.password, salt);
  user
    .save()
    .then(() => {
      return res
        .status(200)
        .json({ success: true, message: "user created successfully", user: user });
    })
    .catch((error) => {
      return res.status(400).json({
        error,
        message: "User not created!",
      });
    });
};
```

```

// Email is verified at backend whether it is already taken or not when client tries to signUp using new Email.
verifyEmail = async (req, res) => {
  await User.findOne({ email: req.body.email }, (err, user) => {
    if (err) {
      return res.status(400).json({ success: false, error: err });
    }
    if (user == null) {
      return res.status(200).json({ success: true });
    }
    return res.status(400).json({ success: false });
  }).catch((err) => console.log(err));
};

// Checking For User authentication when client Tries to Login using Email & Password.
getUser = async (req, res) => {
  const body = req.body;
  const user = await User.findOne({ email: body.email });
  if (user) {
    // check user password with hashed password stored in the database
    const validPassword = await bcrypt.compare(body.password, user.password);
    if (validPassword) {
      res.status(200).json({ success: true, message: "Valid password", user: user });
    } else {
      res.status(400).json({ success: false, message: "Invalid Password", user: null });
    }
  } else {
    res.status(401).json({ success: false, message: "User does not exist", user: null });
  }
};

module.exports = { createUser, getUser, verifyEmail };

```

➤ Movie-controller:

```
const Movie = require('../models/movie-model')
const User = require("../models/user-models");

setRating = (req, res) => {
  const body = req.body

  // body = {title,imdbid,rating,userid,imdbvotes,imdbrating}
  console.log(body);
  const imdbvotes = parseInt(body.imdbvotes.replace(/,/g, ''));
  const rating = parseInt(body.rating);
  const imdbrating = parseFloat(body.imdbrating);
  if (!body) {
    return res.status(400).json({
      success: false,
      error: 'You must provide movie data+rating!!',
    })
  }
  Movie.findOne({ imdbid: body.imdbid }, (err, movie) => {
    if (!movie)// it means movie is not in database so add it to database.
    {
      console.log(body);
      const mov = new Movie();
      mov.title = body.title;
      mov.imdbid = body.imdbid;
      mov.ratecount = imdbvotes + 1;
      mov.totalratesum = parseInt(imdbrating * imdbvotes, 10) + parseInt(rating);
      mov.avgrating = mov.totalratesum / mov.ratecount;
      mov.save().then(() => {
        User.findOne({ _id: body.userid }, (err1, user) => {
          user.ratedmovie.push(body.imdbid)
          user.usererrating.push({ "name":body.title, "rating": rating, "imdbid": body.imdbid })
          user.save().then(() => {
            return res.status(200).json({
              success: true,
              avgrating: mov.avgrating,
              message: 'Movie successfully rated!!',
              user:user,
            })
          })
        })
      })
    }
  })
}
```

```

    })
  })
  .catch(error => {
    return res.status(404).json({
      error,
      message: 'Movie created in database but userratting could not be updated!!',
    })
  })
}
)
})
.catch(error => {
  return res.status(404).json({
    error,
    message: 'Movie could not be created somthing went wrong!!',
  })
})
}
else
{
  User.findOne({ _id: body.userid }, (err1, user) => {
    if (err1) {
      return res.status(404).json({
        error,
        message: 'Something went wrong! user not found!!',
      })
    }
    if (user.ratedmovie.includes(body.imdbid)) {
      const index = user.userratting.findIndex(element => element.imdbid === body.imdbid);
      const oldrating = user.userratting[index].rating;
      user.userratting[index].rating = parseInt(rating);
      user.markModified('userratting');

      user.save().then(() => {
        movie.totalratesum += (parseInt(rating) - oldrating);
        movie.avgratting = movie.totalratesum / movie.ratecount;
        console.log(movie);
        movie.save().then(() => {
          return res.status(200).json({
            success: true,
            avgratting: movie.avgratting,
            message: 'Movie rating successfully updated!!',
            user: user,
          })
        })
      })
      .catch(error => {

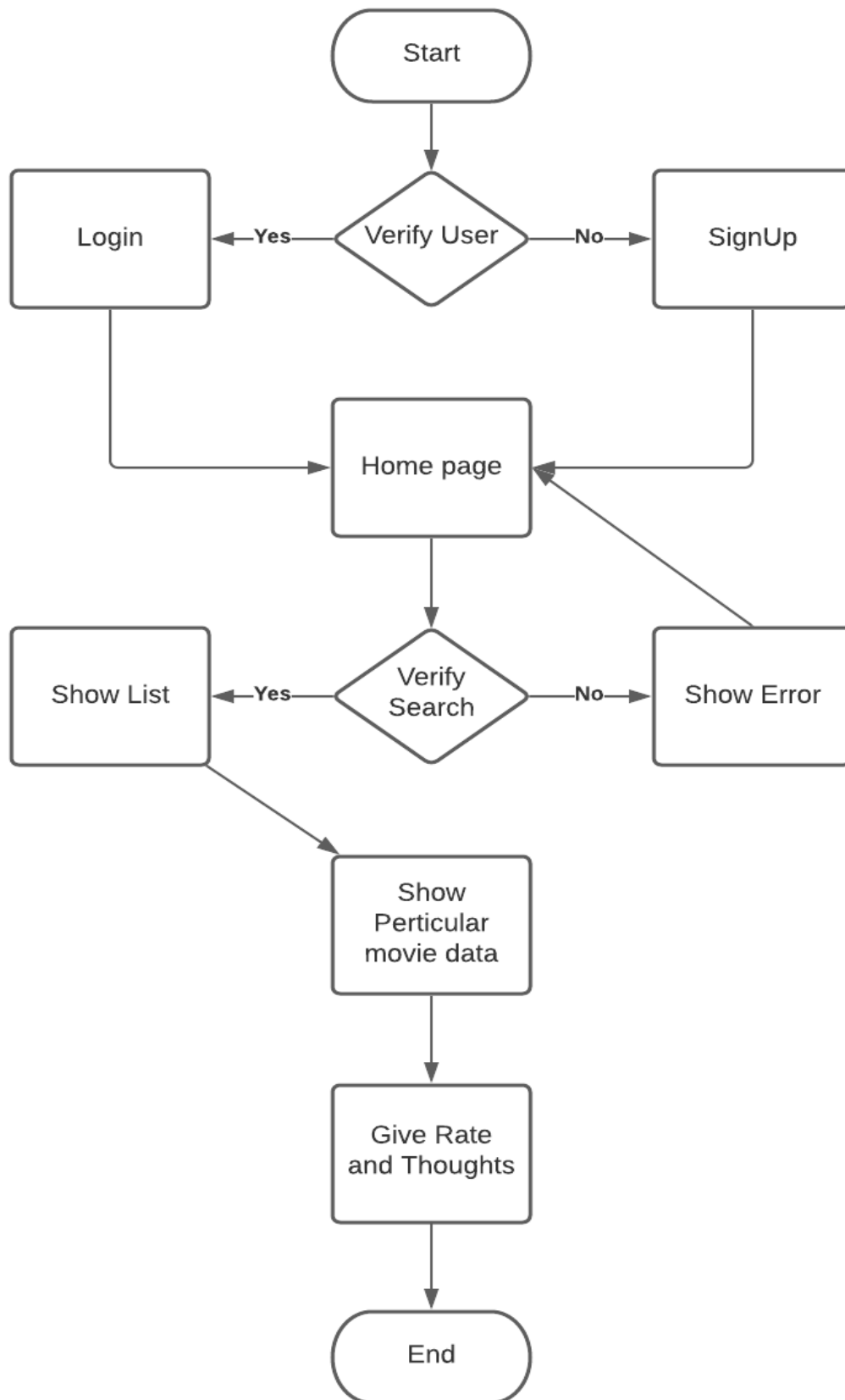
```

```

        return res.status(404).json({
            error,
            message: 'user rating updated but movie rating data can not be updated!!',
        })
    })
}
.catch(error => {
    return res.status(404).json({
        error,
        message: 'Can not update use rating!!',
    })
});
}
else {
    user.ratedmovie.push(body.imdbid)
    user.userratting.push({ "name":body.title,"ratting": ratting, "imdbid": body.imdbid })
    user.save().then(() => {
        movie.totalratesum += parseInt(ratting)
        movie.ratecount++;
        movie.avgratting = movie.totalratesum / movie.ratecount;
        movie.save().then(() => {
            return res.status(200).json({
                success: true,
                avgratting: movie.avgratting,
                newtotalrating:movie.totalratesum,
                message: 'Movie successfully ratted!!',
                user:user,
            })
        })
    })
    .catch(error => {
        return res.status(404).json({
            error,
            message: 'user rating updated bou movie rating data not be updated!',
        })
    })
}
})
.catch(error => {
    return res.status(404).json({
        error,
        message: 'user not found! Movie could not be updated somthing went wrong!!',
    })
})
}
}
}

```


iii. Algorithm/Flowchart:



6. Testing:

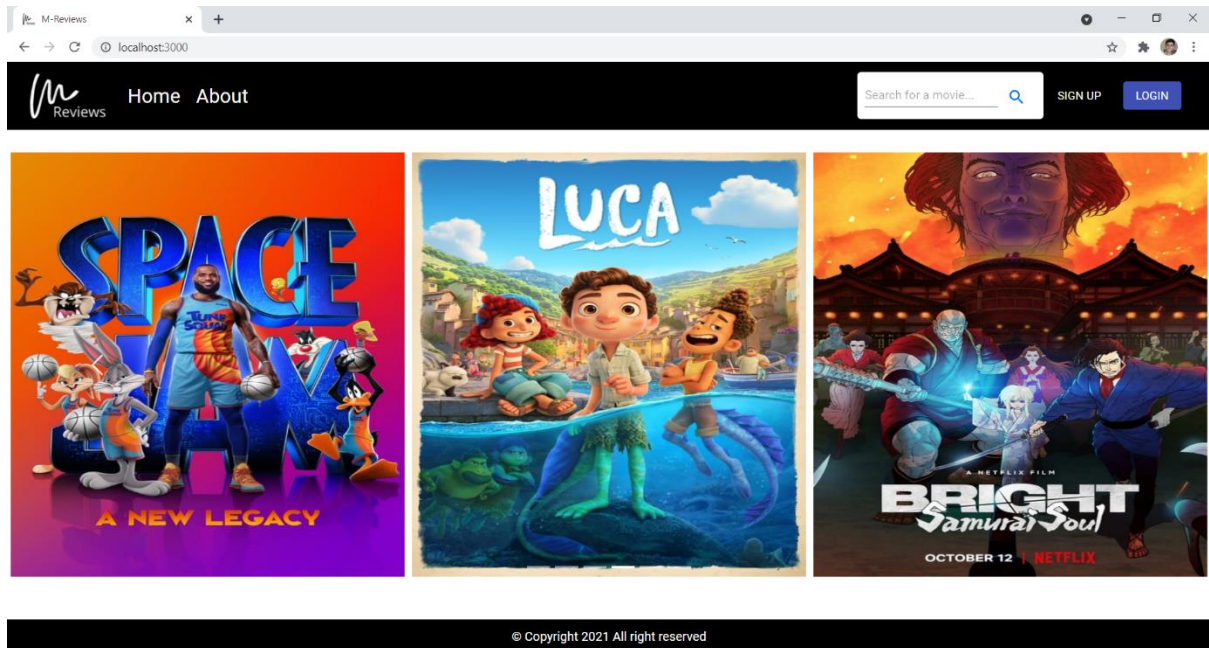
- **Testing Method:** Manual

➤ Manual testing was done during the process of development to make application safer and reduce bugs.

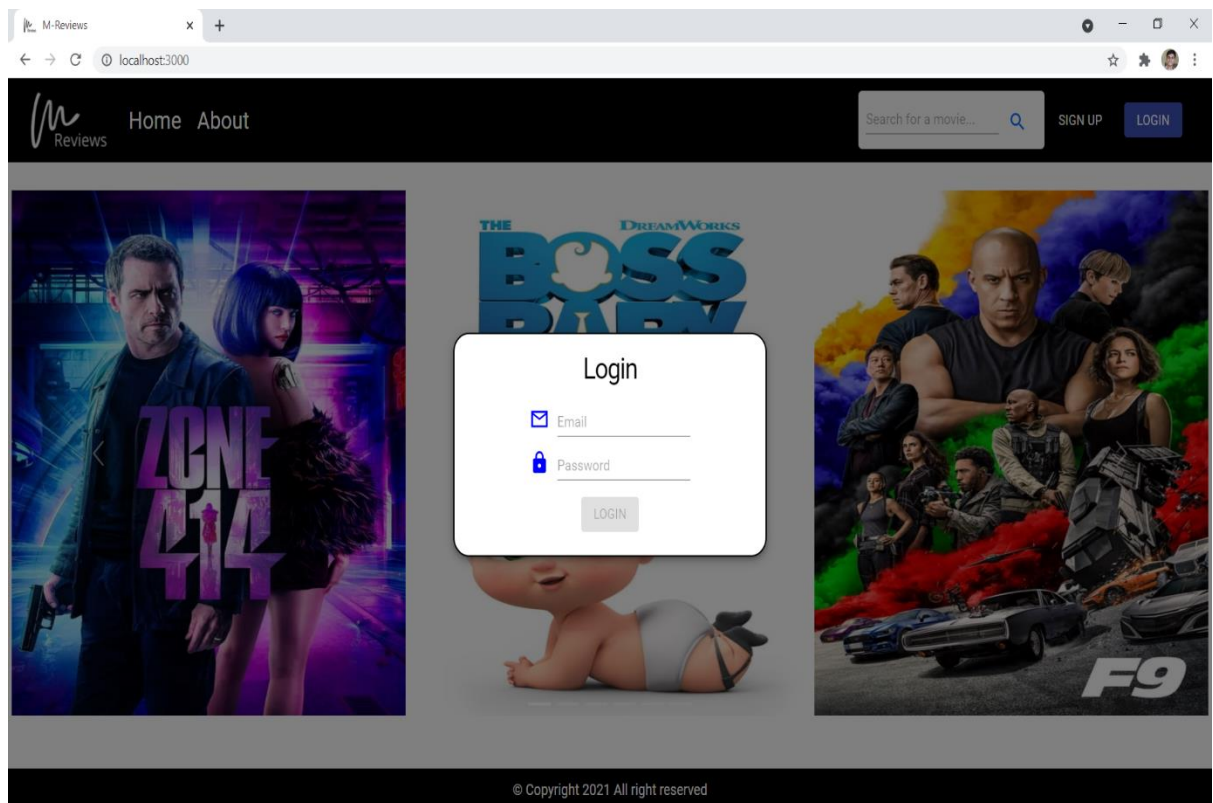
Sr no.	Test Scenario	Expected Result	Actual Result	Status
1	User trying login using invalid credentials	Login Failed with error message 'invalid email or password'	Login Failed	Success
2	User trying to login using correct credentials	User should be logged in	Login Success	Success
3	User trying to signup using existing emailid	User should not allowed to signup	Signup button will not be active	Success
4	User trying to signup using valid fields	User should allowed to login	Signup success	Success
5	User trying to search any movie by entering partial or correct name	Particular or similar kind of movie should be in the result	User is redirected to search page and result was as per need	Success
6	User trying to search movie using gibberish word	Message 'Movie not present! Please enter valid input to search for movie!!'	Able to achieve expected result	Success
7	User trying to search for details of a particular movie after searching it	User should able to see all details like trailer, actor, director, Ratings etc.	User is redirected to another page and able to see all details of a particular movie	Success
8	User trying to rate any movie without login/signup	User should not allowed to rate any movie untill he/she is logged in/signed up	Able to achieve expected result	Success
9	User trying to logout	User should be logged out	Able to achieve Expected result	Success

7. Screenshots:

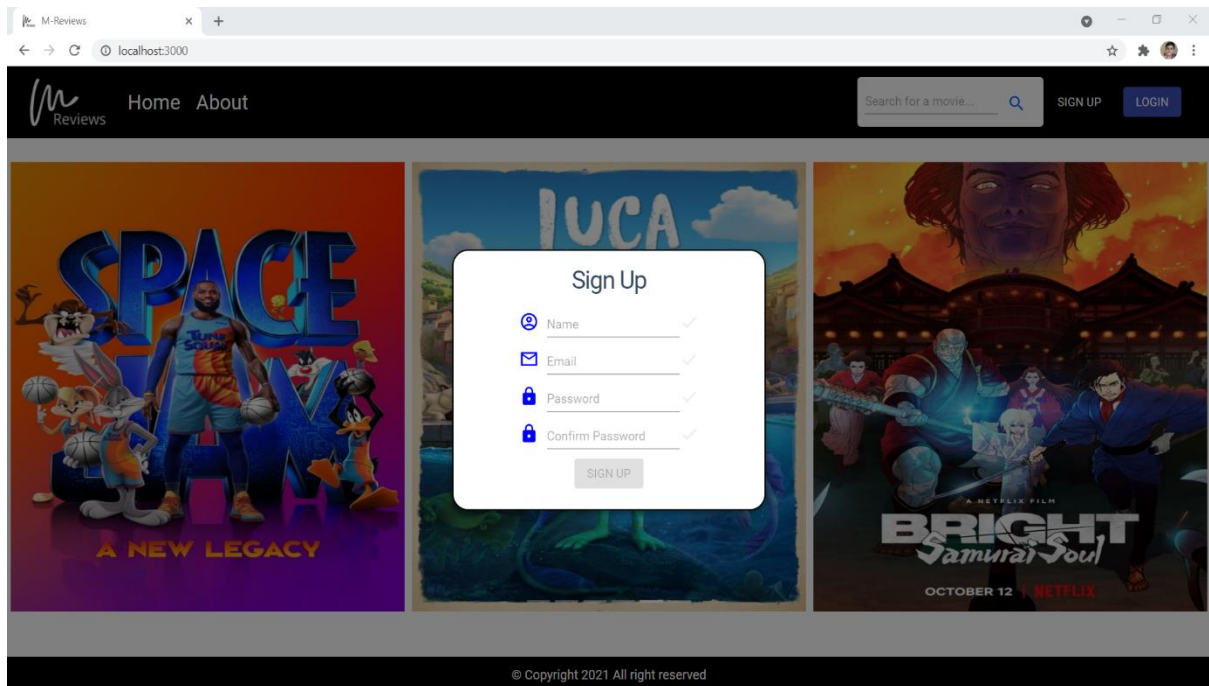
- Home Page without signup/login:



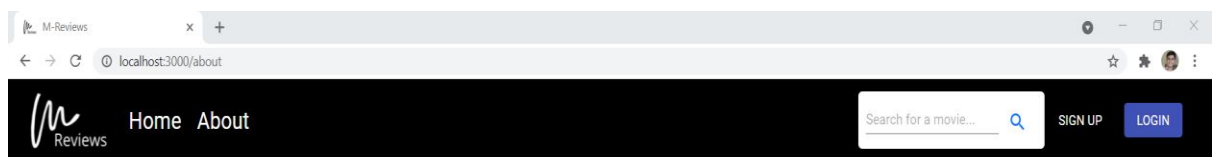
- Login:



• Signup:



• About-us:



About Us

M-Reviews is a online Movie rating and information collection system build to collect data about public interest, from their reviews about movies and Web-Shows.

Data Credits : OMDb and TMDb

Developer Team

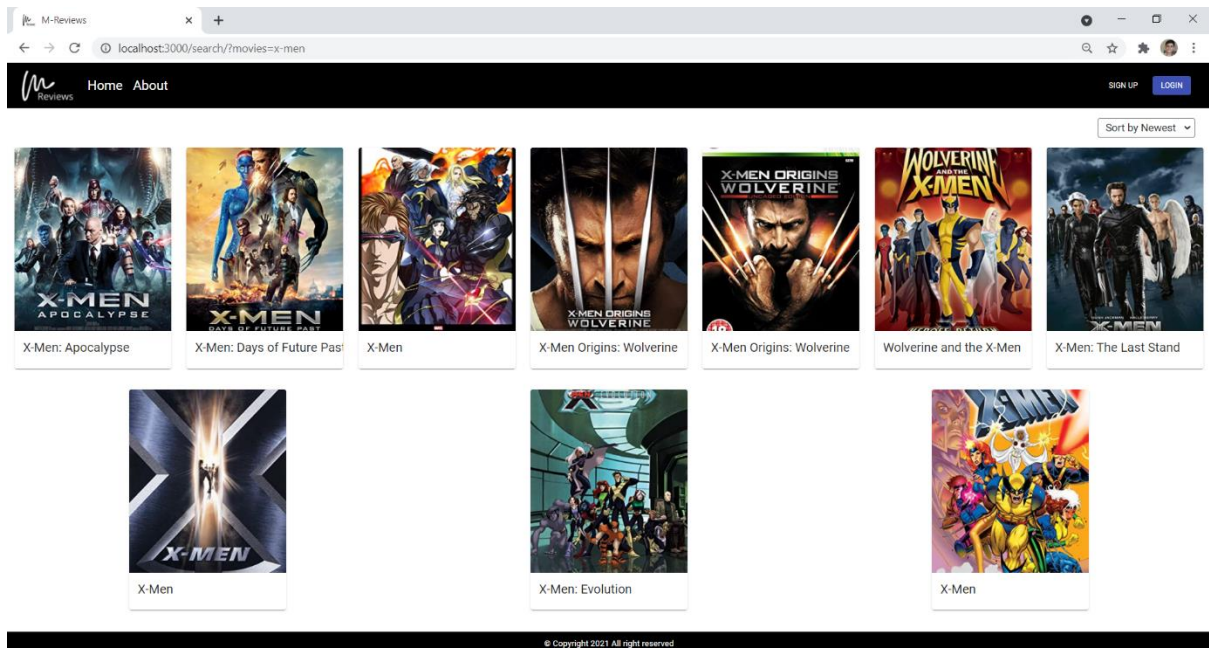
Dhyey Patel
dhyeypatel1612@gmail.com
[Contact](#)

Ronak Padaliya
ronakpadaliya77@gmail.com
[Contact](#)

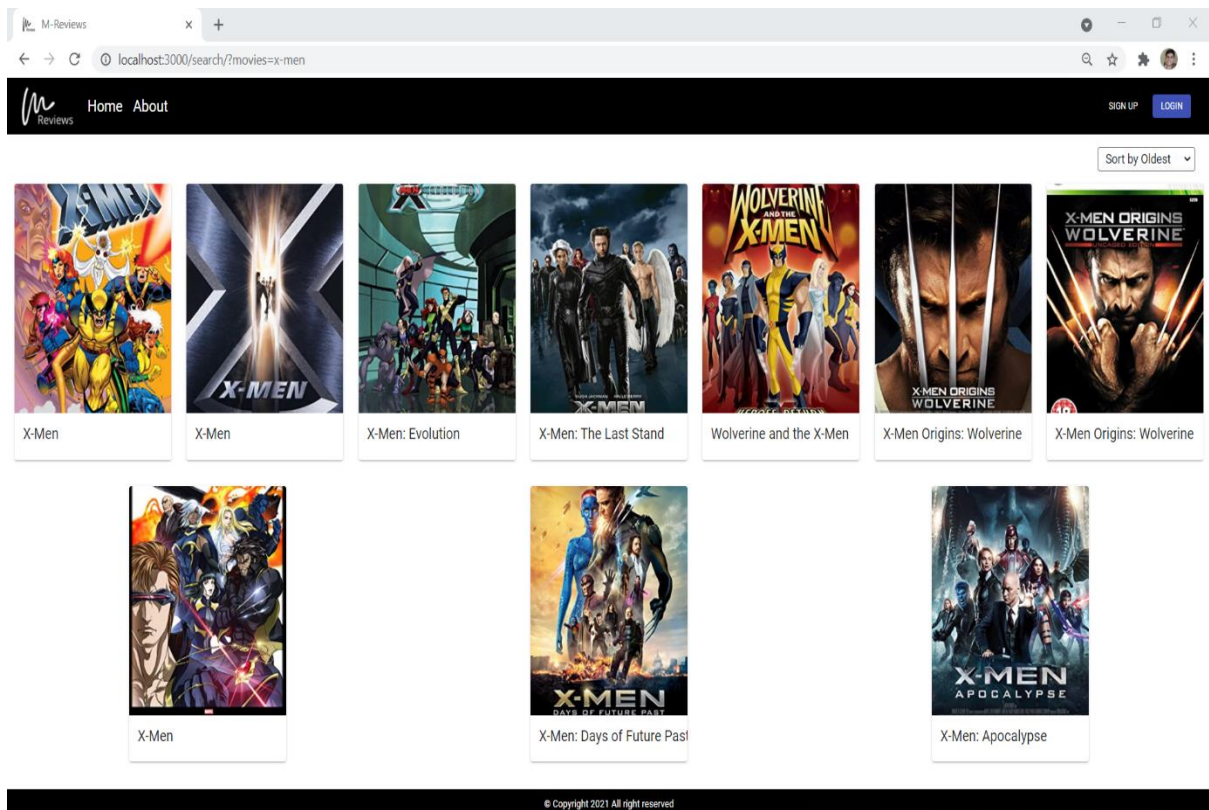
Darshan Parmar
darshanparmar272002@gmail.com
[Contact](#)

© Copyright 2021 All right reserved

- Search Page(sort by Newest):



- Search Page(sort by oldest):



- **Movie-details Page:**

M-Reviews

HomeAbout

SIGN UPLOGIN

X-Men: Apocalypse

movie27 May 2016


Ratings6.9/10

Your Rating-/10

RATE

X-Men: Apocalypse | Teaser Trailer [HD] | 20th Century FOX

Watch laterShare



Watch on YouTube

Story-Line

ActionAdventureSci-Fi

In the 1980s the X-Men must defeat an ancient all-powerful mutant, En Sabah Nur, who intends to thrive through bringing destruction to the world.

Director

Bryan Singer

Actor

James McAvoy, Michael Fassbender, Jennifer Lawrence

Writer/s

Simon Kinberg, Bryan Singer, Michael Dougherty

Awards

1 win & 19 nominations

Collections

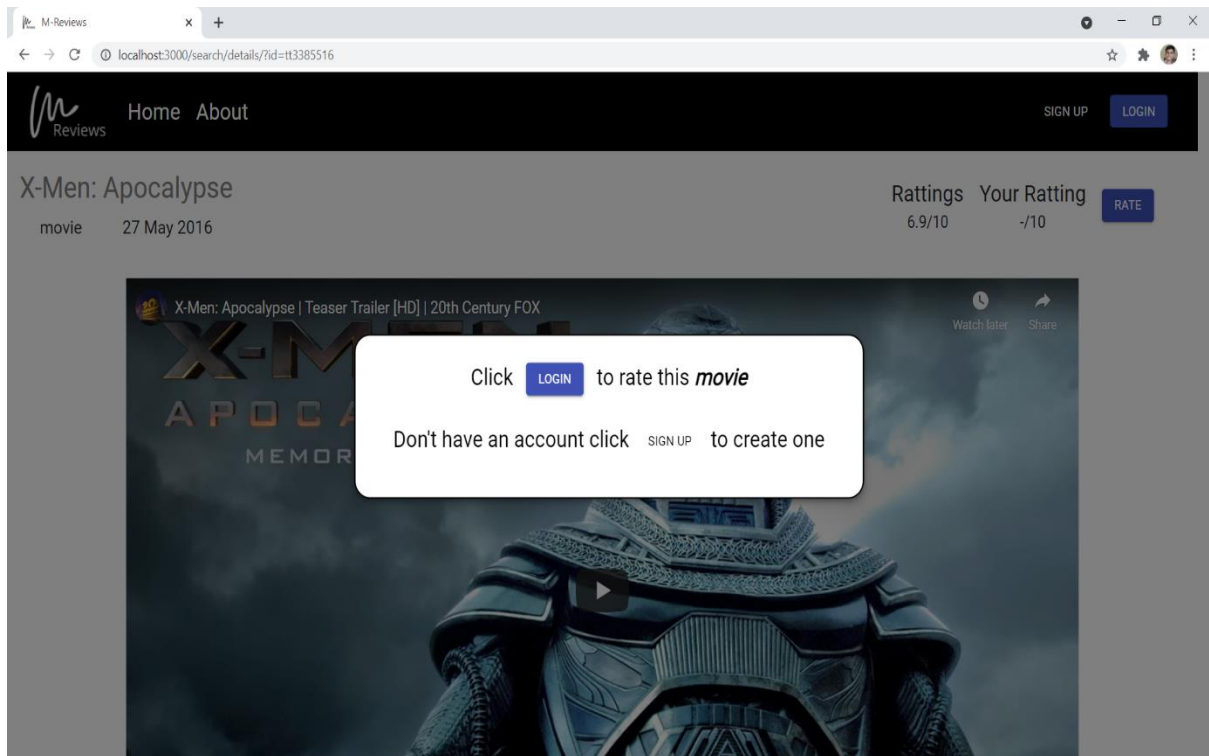
\$155,442,489

Language

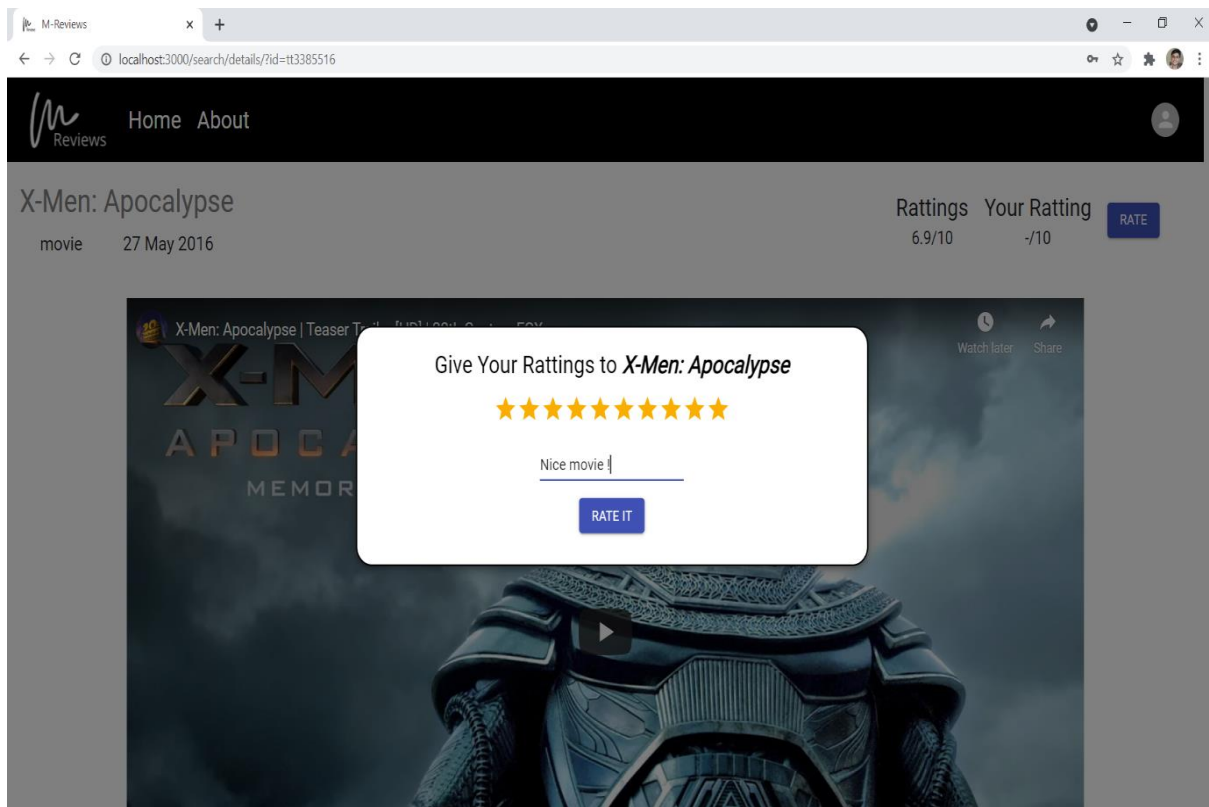
English, Polish, German, Arabic, Egyptian (Ancient)

© Copyright 2021 All right reserved

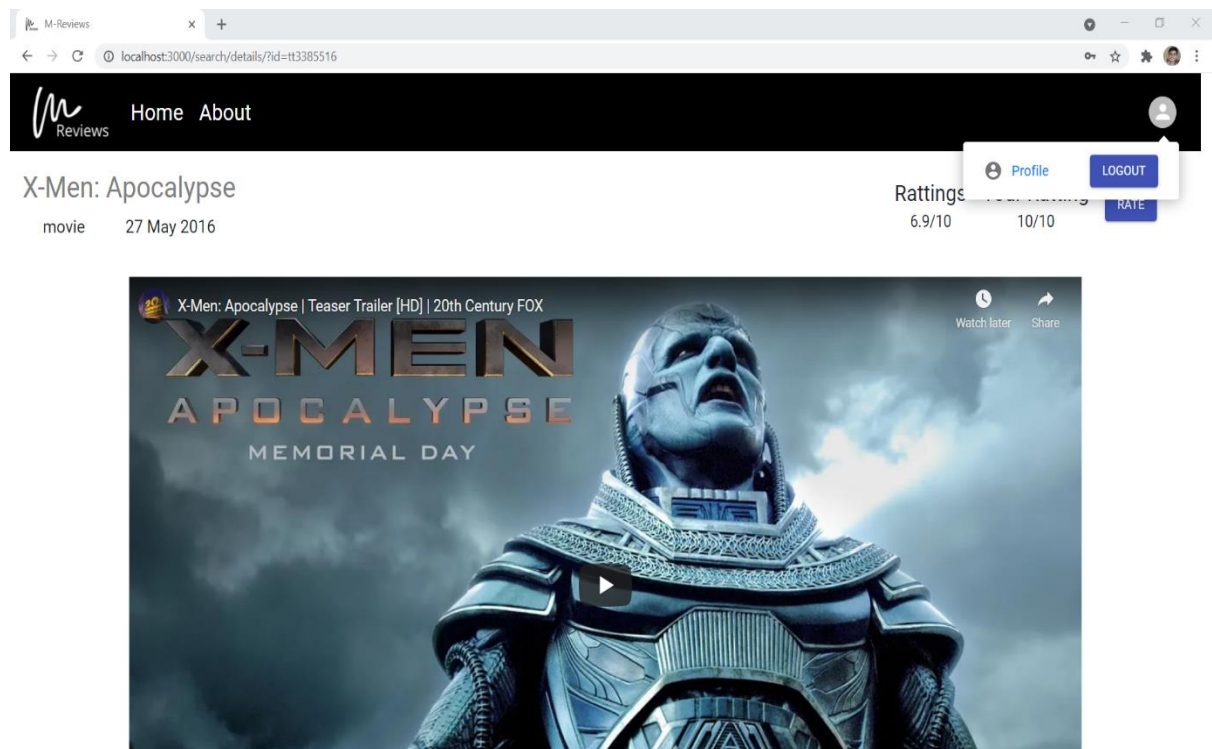
- **Rating movie without Login:**



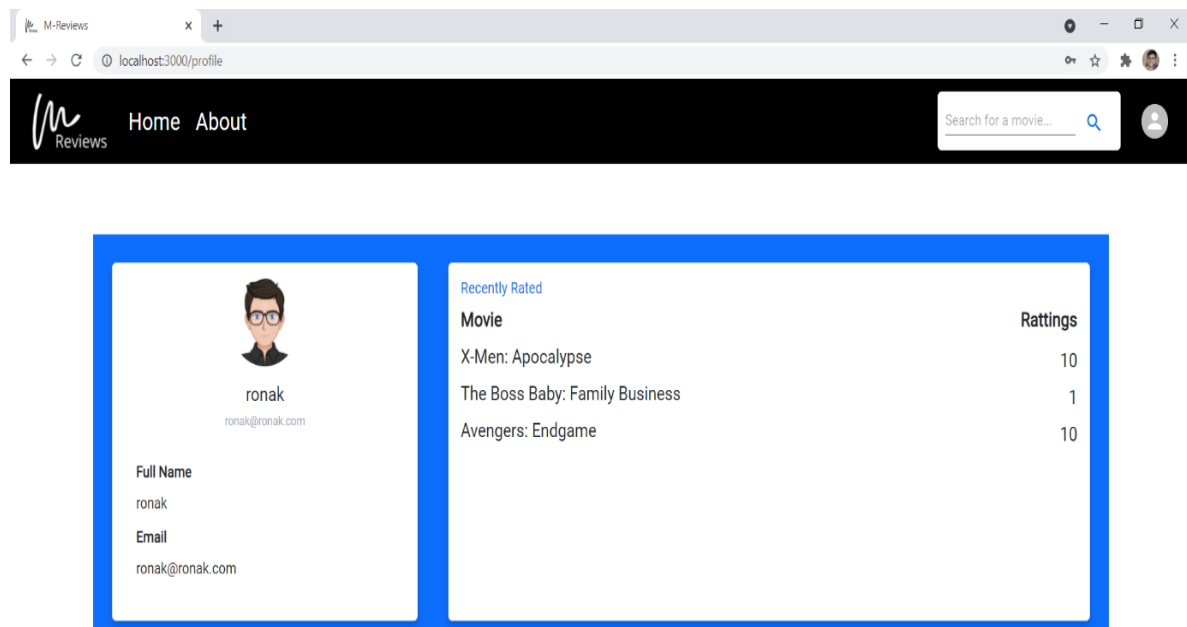
- **Rating movie after login:**



• Account Logo:



• Profile Page:



8. Conclusion:

- The main motive of building this web application is to provide user a good, easy and smooth way of searching a movie and collecting information they needed. They can rate any movie/tv shows they want and give reviews related to it so that it helps other user to personalize their experience. Overall M-Reviews will be helpful to get any related information in just one click.

9. Limitations and future extensions:

- **Limitations:**

- User cannot login if he/she forgets password.
- User cannot update/delete profile.
- Movie can only be searched with proper title or some words of title, no other optimized searching available.
- Some information other than movie or tv-shows like about actors, writers and other supporting cast details are not shown.

- **Future Extensions:**

- Overcome the above limitations.
- Search can be optimized to search movie on the basis of particular fields like language, genre etc.
- Can provide more information related to movie and related stuff.
- Can provide graphical analysis of a particular movie based on rating given by the user.

10.Bibliography:

- stackoverflow.com
- reactjs.org
- nodejs.org
- mongodb.com
- medium.com
- youtube.com/codestepbystep
- omdbapi.com
- themoviedb.org