Neo – Movie

Objective:

Neo Movie is an online application for easily browse and play movies by genres, and language, etc.

Users of the System:

1. Admin
2. User

Functional Requirements:

- Build an application that customer can access the movie.
- Admin should provide for automatic tagging of movie and categorise them.
- Admin should proper movie details such as language, artist, movie name etc..
- User will give the like/dislike for the movie
- This application should have a provision to maintain a database for user information, movie portfolio
- Authenticity for adding users is utmost important for such a website.
- Definitely one should not be allowed to have more than one profile, validation of user should be done using email id.
- **Categorize the movie based on genres**

While the above ones are the basic functional features expected, the below ones can be nice to have add-on features:

- There are lot of freeware & open source applications available for many social functions. The team is expected to search & leverage these to the maximum.
- Multi-factor authentication for the sign-in process

Output/ Post Condition:

- Reports for users
- Reports for Site admin..the operational reports to make the site better in future

Non-Functional Requirements:

Security	• App Platform –UserName/Password-Based Credentials • Sensitive data has to be categorized and stored in a secure manner • Secure connection for transmission of any data
Performance	• Peak Load Performance • Neo Movie-< 3 Sec • Admin application < 2 Sec • Non Peak Load Performance
Availability	• 99.99 % Availability
Standard Features	• Scalability • Maintainability • Usability

	• Availability • Failover
Logging & Auditing	• The system should support logging(app/web/DB) & auditing at all levels
Monitoring	• Should be able to monitor via as-is enterprise monitoring tools
Cloud	• The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure
Browser Compatible	• IE 7+ • Mozilla Firefox Latest – 15 • Google Chrome Latest – 20 • Mobile Ready

Technology Stack

Front End	React Google Material Design Bootstrap / Bulma
Server Side	Spring Boot Spring Web (Rest Controller) Spring Security Spring AOP Spring Hibernate
Core Platform	OpenJDK 11
Database	MySQL or H2

Platform Pre-requisites (Do's and Don'ts):

1. The React app should run in port 8081. Do not run the React app in the port: 3000.
2. Spring boot app should run in port 8080.

Key points to remember:

1. The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.
2. Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.
3. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.
4. Adhere strictly to the endpoints given below.

Application assumptions:

1. The login page should be the first page rendered when the application loads.
2. Manual routing should be restricted by using AuthGaurd by implementing the canActivate interface. For example, if the user enters as http://localhost:3000/signup or http://localhost:3000/home the page should not navigate to the corresponding page instead it should redirect to the login page.
3. Unless logged into the system, the user cannot navigate to any other pages.
4. Logging out must again redirect to the login page.
5. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.
6. Use admin/admin as the username and password to navigate to the admin dashboard.

Validations:

1. Basic email validation should be performed.
2. Basic mobile validation should be performed.

Project Tasks:

API Endpoints:

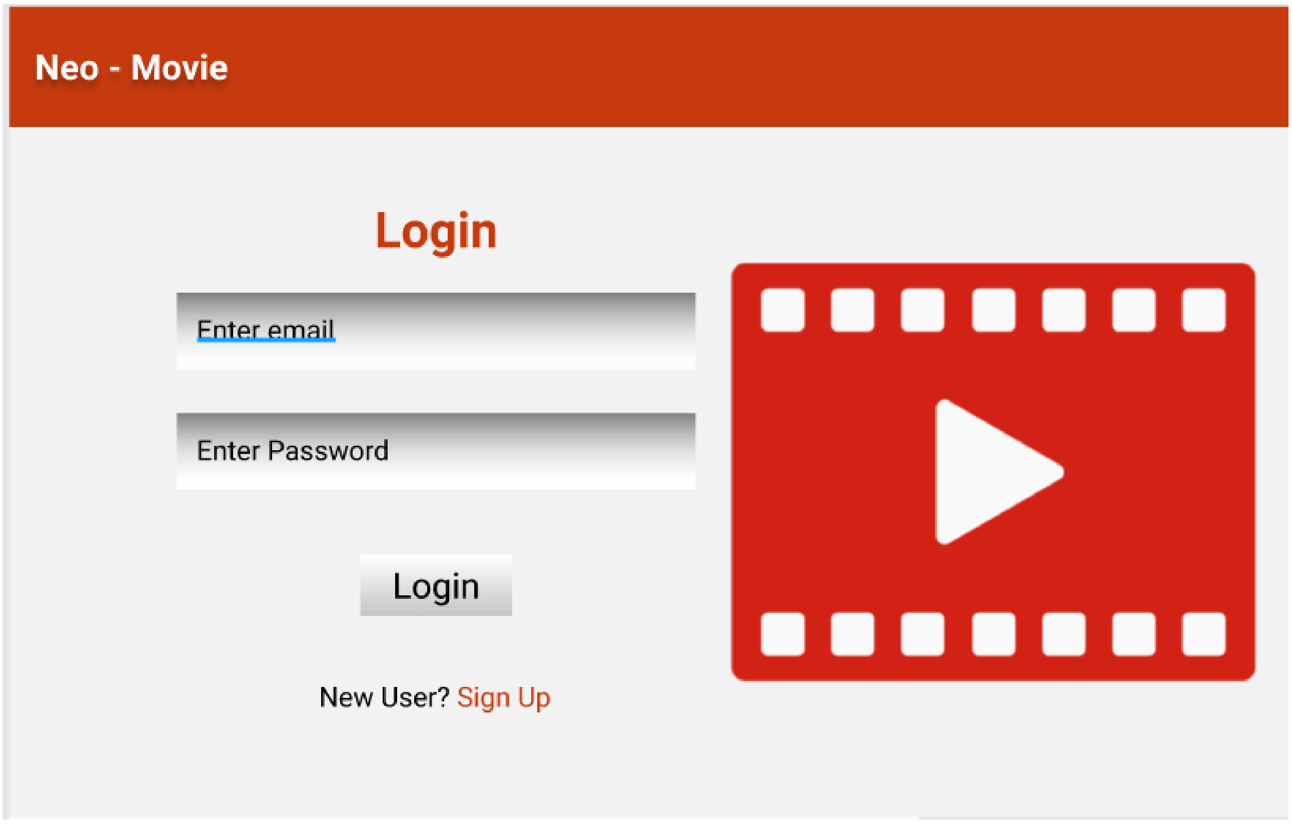
USER			
Action	URL	Method	Response
Login	/login	POST	true/false
Signup	/signup	POST	true/false
Get All Movie	/movie	GET	Array of Movie
Get Movie	/movie/{id}	GET	Movie Details
Add Likes	/like/{id}	POST	Like added to Songs
Remove Likes	/like/{id}	DELETE	Like removed
ADMIN			
Action	URL	Method	Response
Get All User	/admin	GET	Array of Users
Delete User	/admin/delete/{id}	DELETE	User deleted
User Edit	/admin/userEdit/{id}	PUT	Save the Changes
Get All Movie	/admin/movie	GET	Array of Movie
Delete Movie	/admin/movie/{id}	DELETE	Movie deleted
Update Movie	/admin/movie/{id}	PUT	Save the Changes
Get All Comment	/admin/comment	GET	Array of Comments

Frontend:

User:

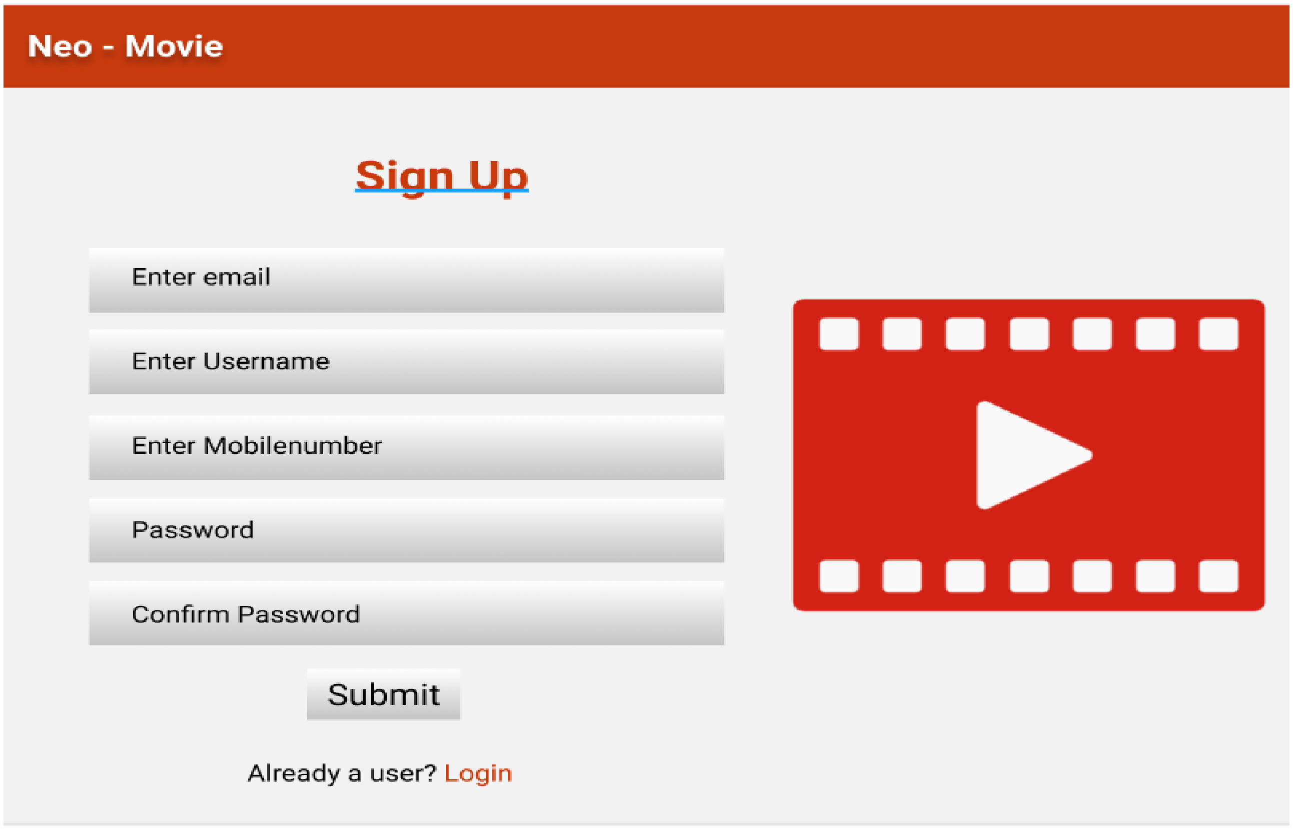
Login:

Output Sreenshot:



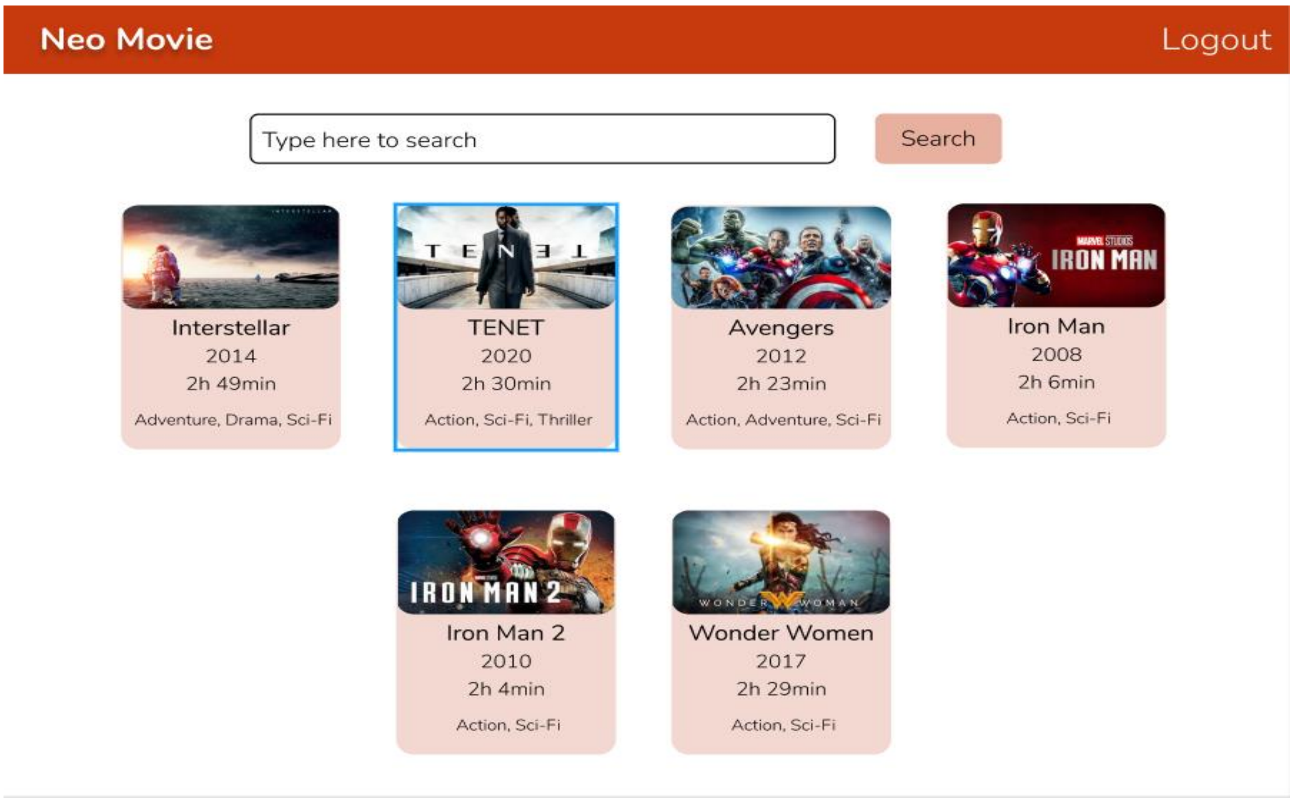
Signup:

Output Screenshot:



Home:

Output Screenshot:



Interstellar
2014
2h 49min
Adventure, Drama, Sci-Fi

TENET
2020
2h 30min
Action, Sci-Fi, Thriller

Avengers
2012
2h 23min
Action, Adventure, Sci-Fi

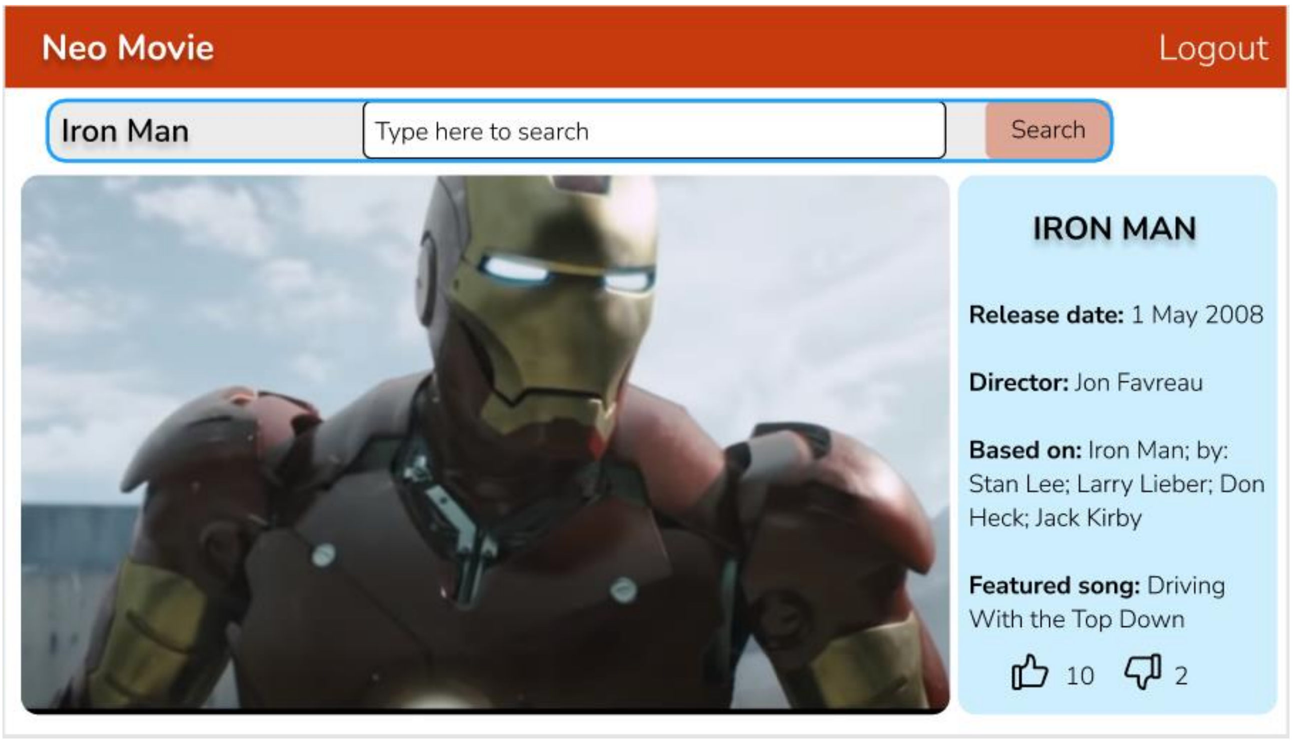
Iron Man
2008
2h 6min
Action, Sci-Fi

Iron Man 2
2010
2h 4min
Action, Sci-Fi

Wonder Women
2017
2h 29min
Action, Sci-Fi

Movie:

Output Screenshot:



IRON MAN

Release date: 1 May 2008

Director: Jon Favreau

Based on: Iron Man; by: Stan Lee; Larry Lieber; Don Heck; Jack Kirby

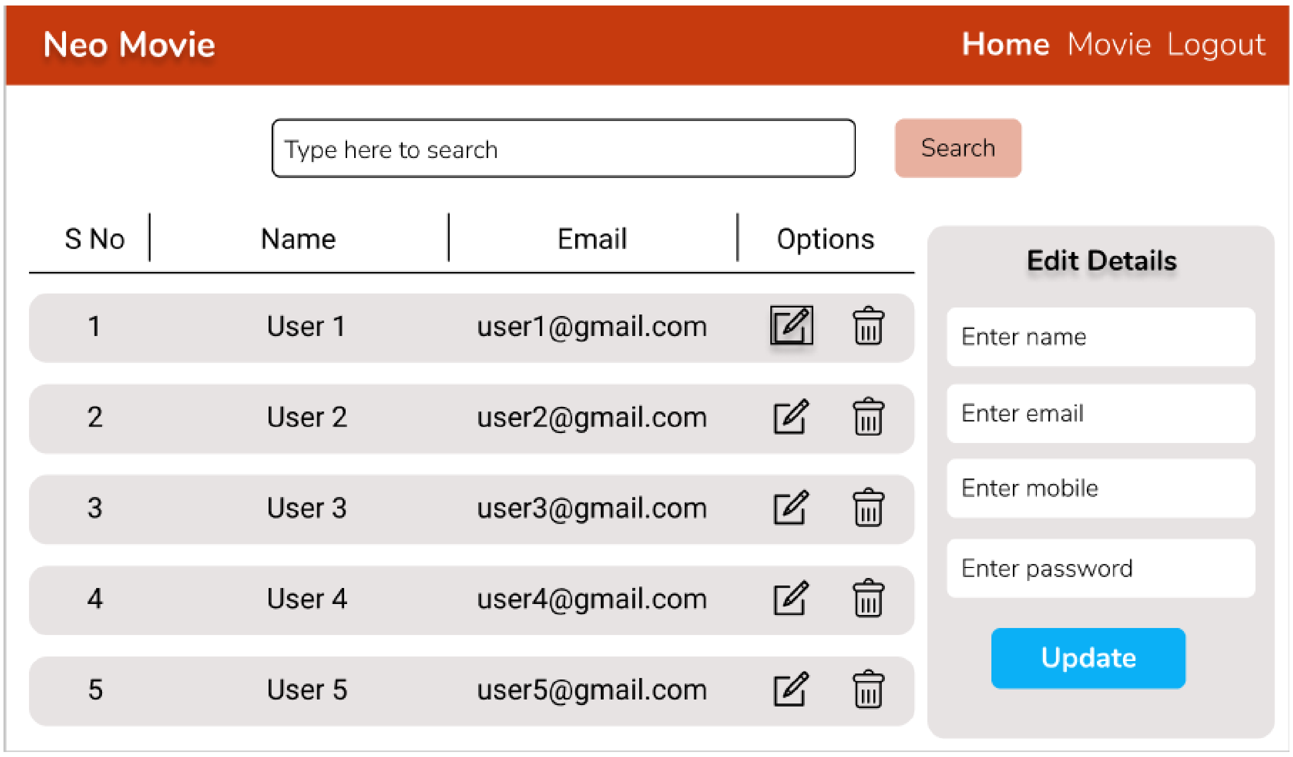
Featured song: Driving With the Top Down

👍 10 👎 2

<u>Admin:</u>

Home:

Output Screenshot:



Neo Movie

Home Movie Logout

Type here to search

Search

S No	Name	Email	Options
1	User 1	user1@gmail.com	✎ 🗑
2	User 2	user2@gmail.com	✎ 🗑
3	User 3	user3@gmail.com	✎ 🗑
4	User 4	user4@gmail.com	✎ 🗑
5	User 5	user5@gmail.com	✎ 🗑

Edit Details

Enter name

Enter email

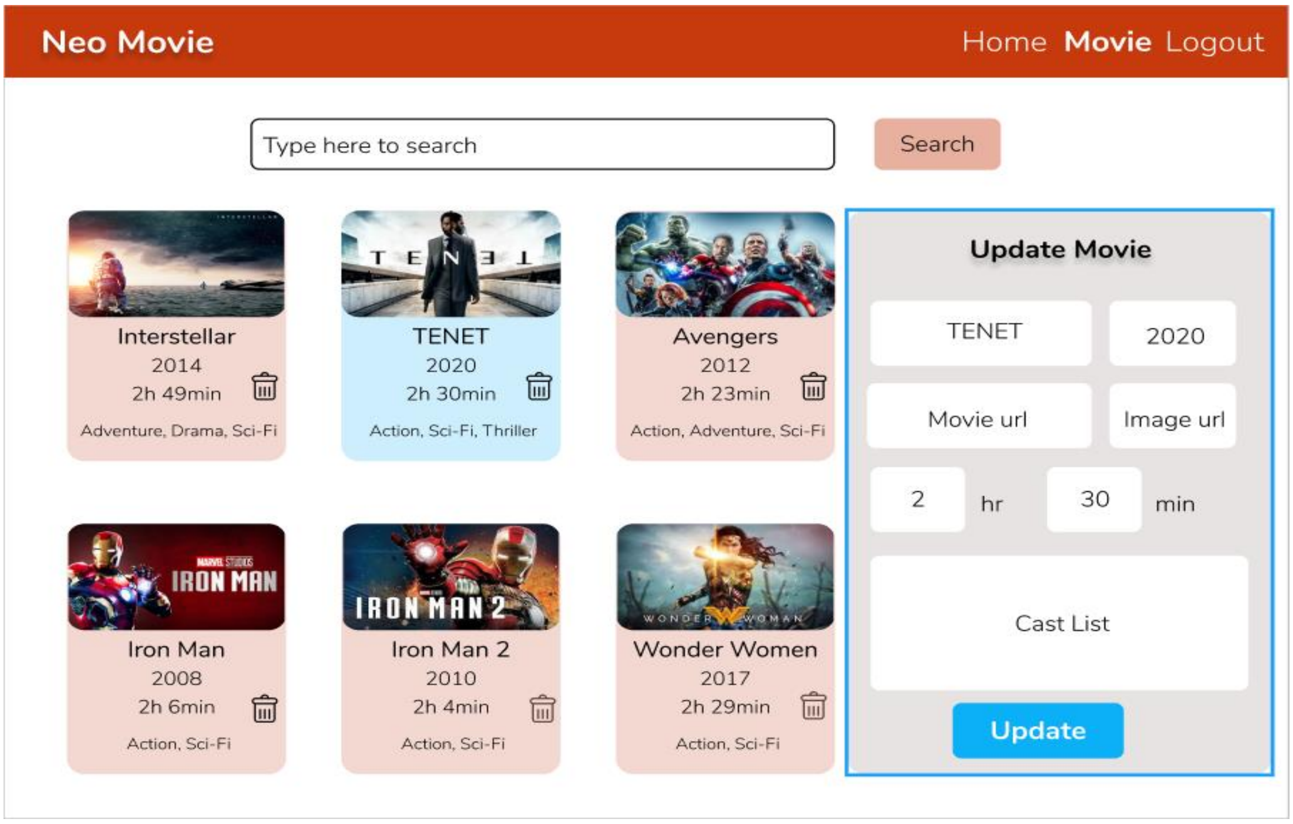
Enter mobile

Enter password

Update

Movie:

Output Screenshot:



Neo Movie

Home **Movie** Logout

Type here to search

Search

Interstellar
2014
2h 49min
Adventure, Drama, Sci-Fi

TENET
2020
2h 30min
Action, Sci-Fi, Thriller

Avengers
2012
2h 23min
Action, Adventure, Sci-Fi

Iron Man
2008
2h 6min
Action, Sci-Fi

Iron Man 2
2010
2h 4min
Action, Sci-Fi

Wonder Women
2017
2h 29min
Action, Sci-Fi

Update Movie

TENET | 2020

Movie url | Image url

2 hr 30 min

Cast List

Update

<u>Backend:</u>

<u>Class and Method description:</u>

<u>Model Layer:</u>

1. UserModel: This class stores the user type (admin or user) and all user information.
    a. Attributes:
        i. email: String
        ii. password: String
        iii. username: String
        iv. mobileNumber: String
        v. active: Boolean
        vi. role: String
    b. Methods: -

2. LoginModel: This class contains the email and password of the user.
    a. Attributes:
        i. email: String
        ii. password: String
    b. Methods: -
3. MovieModel: This class stores the details of the product.
    a. Attributes:
        i. movieId: String
        ii. movieName: String
        iii. movieUrl: String
        iv. moviePosterUrl: String
        v. movieCast: List<String>
        vi. like: LikeModel
    b. Methods: -
4. LikeModel: This class stores the cart items.
    a. Attributes:
        i. Id: String
        ii. noOfLike: int

iii. likedUser: List<UserModel>

b. Methods: -

Controller Layer:

5. SignupController: This class control the user signup
    a. Attributes: -
    b. Methods:
        i. saveUser(UserModel user): This method helps to store users in the database and return true or false based on the database transaction.
6. LoginController: This class controls the user login.
    a. Attributes: -
    b. Methods:
        i. checkUser(LoginModel data): This method helps the user to sign up for the application and must return true or false
7. UserController: This class controls the add/edit/update/view User.
    a. Attributes: -
    b. Methods:
        i. List<UserModel> getUser(): This method helps the admin to fetch all User from the database.
        ii. List< UserModel > getOnlineUser(): This method helps to retrieve all the online user from the database.
        iii. userModel userEditData(String id): This method helps to retrieve a user from the database based on the user id.
        iv. userEditSave(UserModel data): This method helps to edit a user and save it to the database.
        v. userDelete (String id): This method helps to delete a user from the database.
8. MovieController: This class helps in adding movie, deletingmovie, updating updating.
    a. Attributes: -
    b. Methods:
        i. addMovie(MovieModel image): This method helps the user to add the movie to the database.
        ii. List<MovieModel> getAllMovie(): This method helps the user to fetch all the movie from the database.
        iii. MovieModel showMovie(String id): This method helps the user to play the movie.

iv. deleteMovie(String id): This method helps to delete a movie from the database.

v. updateMovie(MovieModel data): This method helps to update a movie details from the database.

9. CommentController: This class helps in adding, deleting, updating the comment.
    a. Attributes: -
    b. Methods:
        i. addLike(String Id): This method helps the user to add the Like for the selected song.
        ii. removeLive (String id): This method helps to remove a Like from the song.
        iii. getLikeCount(String id): This method will return the number likes based on song id.