Movie Recommender System : Final Project Report

Team Members:

180050009 : Anjali Priya
180050027 : C.Shreya
180050064 : Muthyala Leela Sri Pragna
180050065 : Nancy Yadav
180050104 : Sindhu Pawar

Requirements (Enumerated):

There are three roles - user,editor,admin.

Users can search for a movie by movie name,actor/actress name or by genre in the home page.Users can send friend requests ,accept request and deny request from other users by using accept request,cancel request buttons respectively.User can add movies to watchlist,watching now pages and also can remove from watchlist and watching .Movies which are being currently watched will be in watching now and user can rate the movies in watching now.Users who are friends can recommend movies to each other. There is a profile page in which user can edit his profile like change name or gender or country etc.There is a friends page in which all the user's friends will be displayed and the user can recommend the movies to his/her friends via recommend button.On clicking the recommend button it will redirect to the recommend page in which there are movies the user can recommend.Movies recommended by friends will appear in the "movies recommended by friends" page.In home page movies are recommended to the users based on the age,country and the language.User can add and remove actor/actress to his/her favourites via Add to favourite button/Remove from favourites button respectively. All the movies starring the favourite actors/actresses of user's will be displayed in favourites page.Editors can add movies,add actors/actresses and language to the movies.Admins can do everything editors and users can do.

Use case descriptions:

1. Registration Page:
   ➔ *Use cases on this page: "creating an account" button, "Sign-in" button*
   - "Creating an account" button - After entering the details on clicking this button user can create a account and this

- "Sign in" button - If a user already has an account he/she can click on this on this button to login.

2. Login Page:
   ➔ *Use cases on this page: "login button"*
   - The user has to click the "login" button after entering the username and password to log in to the account.

3. Home Page:
   ➔ *Use cases on this page: "Add to Watch later", "View detail", "View movie", "logout", "Movie search engine", "Actor/Actress search engine", "Genre search engine"*
   - It contains three search engines one for searching the movies by movie name and second one for searching the movies by cast/crew(like actors, actresses, director) and third one for searching the movies by genre.
   - All movies recommended to the user are displayed here. For each movie, there are add to watch later, view movie, view detail buttons.
   - Clicking on the "Watching" button will add the movie to the "Watching now" page and clicking on the "Add to watchlist " button will add the movie to the "Watchlist" page.
   - It also contains the "logout" button to logout of the account.

4. Watchlist Page:
   ➔ *Use cases on this page: "View detail", "View movie", "Remove from watchlist"*
   - "View detail and View movie" is the same as in the home page.
   - "Remove from watch later" : Removes the movie from the watchlist page.

5. Watching now Page:
   ➔ *Use cases on this page: "View detail", "Rate movie"(Like/Dislike and point-based Rating),"Remove from Watching now"*
   - On clicking "View detail" button, we get the details of the movie
   - This page contains all movies that are being watched now by clicking the "View movie" button on any page that contains this button.
   - "Rate movie" button is to like or dislike the movie and rate the movie.
   - If we do not wish to watch a movie, it can be removed by "Remove from Watching now" button.

6. Friends Page:
   ➔ *Use cases on this page: "Recommend a movie", "Send Request", " Cancel Request".*
   - This page contains a search engine for searching for other users by username. If you wish to add someone as your friend then you can search for them by their username.

- If the user is already your friend then it displays “You are friends with this user” else if you have already sent the request then it displays the “Cancel request” button otherwise it displays the “Send request” button
- It displays all friends of the user and each friend will have the “Recommend a movie” button by which we can recommend a movie by the movie name to someone.

7. Friend Request Page:
   ➔ *Use cases on this page: “Delete Request”, ”Accept Request”.*
   - This page contains all the friend requests to the user from other users.
   - Each request is displayed by the user name and name of the person sending this request with delete request and accept request buttons.

8. Movies Recommended by friends page:
   ➔ *Use cases on this page: “Add to watch later”, ”View movie”.*
   - This page contains all the movies recommended by a user's friends.
   - For each movie, there are add to watch later, view movie, view detail buttons and the username of the person who recommended the movie.

9. Profile Page:
   ➔ *Use cases on this page: “Edit Profile” , ”Add Language”, “Remove Language”*
   - This page contains details filled in by the user during registration and the “Edit Profile” button to edit these details.
   - If the admin gave edit access to the user then it displays you have edit access. (Edit access meaning the user is an editor and thus can add or remove movies)
   - It contains the “Add language ” and “Remove language” buttons to add and remove languages from the list of languages that a user has chosen to watch movies.

10. Favourite Page:
    ➔ *Use Cases on this page: “Add to favourite”, ”Remove from favorite”, ”Actor/Actress search engine”.*
    - This page contains all the favourite actors/actresses of the user and also a search engine to search the actors/actresses by their name.
    - After searching an actor/actress by their name, it shows the actors/actresses the user searched and if the actor/actress is already added to favorites then it will show a “Remove from favourite” button or else it will show “Add to favourite” button.
    - If a new movie is added to the database in which the user's favourite actor/actress acted, then a message pops saying that ‘a new movie is present in which your favourite actor/actress acted’.

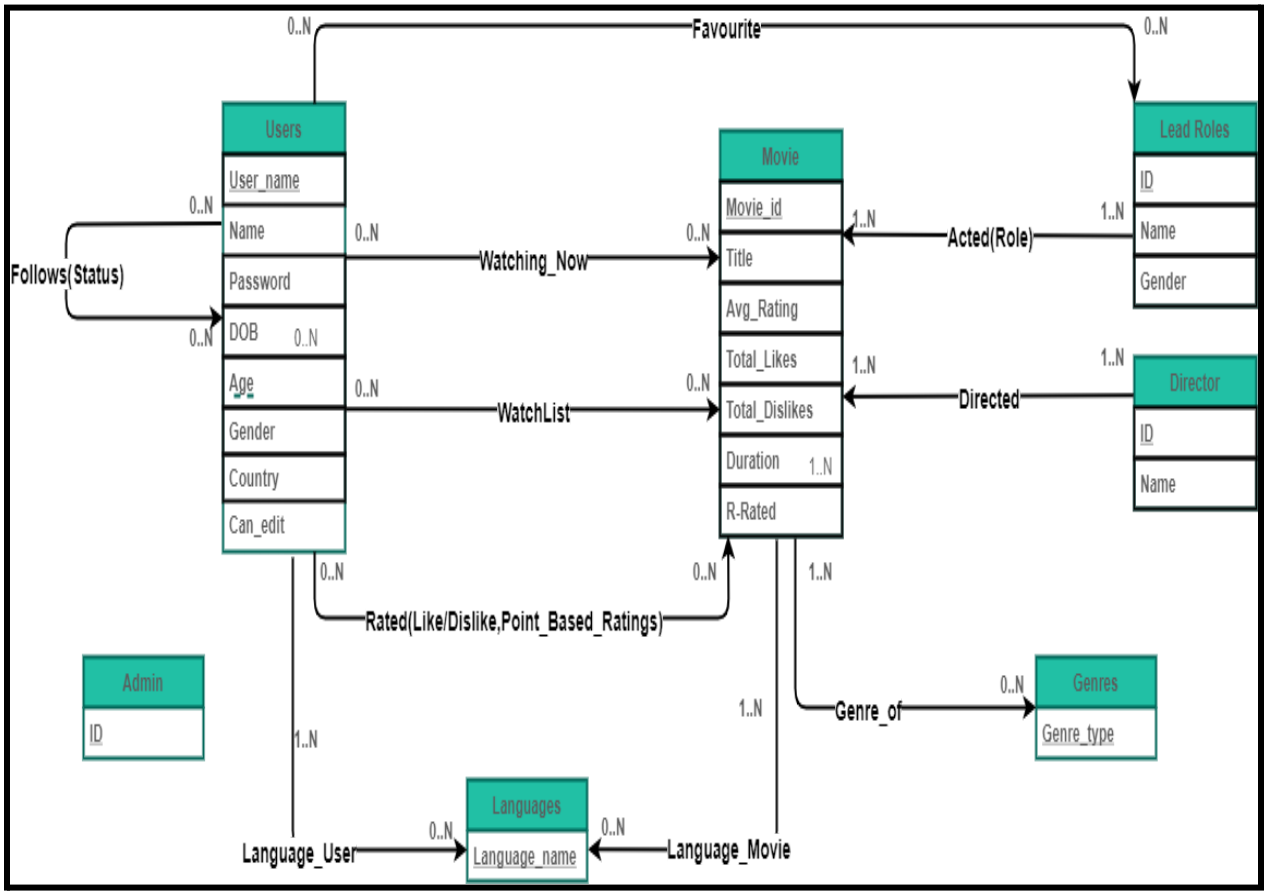
11. Add Movie Page: (This page is only accessed by editors and admin)
    - ➔ *Use cases on this page: "Add movie", " Add actor/actress", " Add director", "Add genre"*
        - "Add movie" button adds a movie and the details of the movie to the database.
        - This page also contains "Add actor/actress", " Add director", "Add genre" buttons to add the movie to the database. Editor/Admin can add actor/actress or director or genre only if the movie is already present in the database if not it has to be added by using the "Add movie" button first.

12. Remove Movie Page: (This page is only accessed by editors and admin)
    - ➔ *Use cases on this page: "Remove movie"*
        - This page contains a "Remove movie" button to remove the movie from the database. It is removed only if the movie is already present in the database.

13. Admin Page: (This page is only accessed by admin)
    - ➔ *Use cases on this page: "Remove users", "Give Edit Access"*
        - This page contains a "Remove users" button to remove the user from the database
        - The "Give edit access" button gives edit access to users so that a user can become an editor.

Design:

Normalized Schema:



Constraints & Triggers:

1.Users :-

- Constraints:
  - User_name, Name, Password, DOB, Age, Country, Can_edit should not be null.
  - Gender can be null.
- Triggers:
  - Can_edit: It will be updated when the admin allows any user the edit access.

2.Movie:-

- Constraints:

    - Movie_id, Title, Duration, R-Rated should not be null.
    - Title is Unique.
    - Initially, Total_likes, Total_dislikes & Avg_Rating should be 0.
- Triggers:
    - Whenever a movie is rated or liked or disliked, the Avg_Rating, Total_likes, Total_dislikes columns will be updated for that movie.

3.Lead Roles:-

- Constraints:
    - ID, Name, Gender should not be null.
    - Name is Unique.
- Triggers:
    - Whenever a new movie is added in Movie Table, the columns ID, Name, Gender in this table will be updated with the details of the lead two casts.

4.Director:-

- Constraints:
    - ID, Name should not be null.
    - Name is Unique.
- Triggers:
    - Whenever a new movie is added in the Movie Table, the columns ID, Name, Gender in this table will be updated with the details of the Director.

5.Languages:-

- Constraints:
    - Language_name should not be null.
- Triggers:
    - Whenever a new movie is added in Movie Table, the Language_name in this table will be updated with the language of the movie if the language of the new movie does not already exist in the column.

6.Genres:-

- Constraints:
    - Genre_type should not be null.
- Triggers:
    - Whenever a new movie is added in Movie Table, the Genre_type in this table will be updated with the Genre of the movie if the Genre of the new movie does not already exist in the column.

7.Admin:-

- Constraints:
    - ID should not be null.

INDEXING:

1.Users :-

- User_name:
    - Type of index : Collaborative (as User_name is primary key).
    - Reason: Default index.

2.Movie :-

- Movie_id:
    - Type of index : Collaborative (as Movie_id is a primary key).
    - Reason: Default index.
- Avg_Rating:
    - Type of index : Non-collaborative in descending order (Avg_Rating is not a primary key).
    - Reason: We use select query very frequently on this column when we try to recommend based on Avg_Rating . So We keep index for this column as the cost for adding index and querying on indexed column is quite less then directly querying the database. Also there is a maximum one update in Avg_rating per user per movie but here the indexing on Movie_id will be used which is any way the primary key.

NOTE: When we add a new movie in Movie Table there is only one update which is not very less frequent compared to the select queries on this table.

3.Lead Roles :-

- ID:
    - Type of index : Collaborative (as ID is a primary key).
    - Reason: Default index.
- Name :
    - Type of index : Non- Collaborative (as Name is not a primary key).
    - Reason: We use select query very frequently on this column when we try to recommend based on Actors worked in movies which are already watched by the user . So We keep index for this column as the cost for adding index and querying on indexed column is quite less then directly querying the database.

4.Director :-

- ID:
    - Type of index : Collaborative (as ID is a primary key).
    - Reason: Default index.
- Name :

- Type of index : Non- Collaborative (as Name is not a primary key).
- Reason: We use select query very frequently on this column when we try to recommend based on Directors who directed the movies already watched by the user. So We keep index for this column as the cost for adding index and querying on indexed column is quite less then directly querying the database.

5.Languages :-

- Language_name**:**
  - Type of index : Collaborative (as Language_name is a primary key).
  - Reason: Default index.

6.Genres :-

- Genre_type**:**
  - Type of index : Collaborative (as Genre_type is a primary key).
  - Reason: Default index.

7.Admin :-

- ID**:**
  - Type of index : Collaborative (as ID is a primary key).
  - Reason: Default index.

Test cases:

In Registration Page:

1. In this test a user's account will be created with a unique Username.
   - ➔ Requirement it is testing : Username should be unique (not null) and password should contain at least one special character and digit and lowercase alphabets.
   - ➔ Use case it is testing : Click on Create Account button
   - ➔ Inputs to the test :
     - ◆ CORRECT INPUT : Username, Name, DATE_of_Birth, Country, Gender, Password.
       ex- Nancy, 22/04/2000, India, Female, XYZ
     - ◆ INCORRECT INPUT : Username that already exists in Users table,
       ex- Nancy22 , 23/01/2001, India, Female, XYZ
   - ➔ Expected outputs : No output
   - ➔ Expected backend changes : Adding the user details to the table Users.
   - ➔ Test procedure : It will redirect to the login page if the username is correct else if the username is null or incorrect or already in use it will redirect to the registration page & “The Username Nancy22 already exists" will be displayed.

In Login Page:

1.
   - ➔ Requirement it is testing : Username and Password should be correct.
   - ➔ Use case it is testing : Click on login button after giving required credentials
   - ➔ Inputs to the test
     - ◆ Correct input :Correct User_name & Correct Password . Ex- Nancy22, XYZ

- ◆ Incorrect input :Correct User_name & Incorrect Password . Ex- Nancy22, ABC

➔ Expected outputs : No output.
➔ Expected backend changes : No change.
➔ Test procedure : It is directed to the Home page if the username and password pair exists in our database else if password is incorrect then it will redirect to login page displaying password is not correct.

In Home Page:

1.
➔ Requirement it is testing : On clicking the home button, popular movies must be displayed.
➔ Use case it is testing : Click Home button
➔ Inputs to the test : No input
➔ Expected outputs : Movie List will be displayed like this
★ Movie name : “Toy Story”
★ View detail button
★ View movie button
★ Add to watchlist button
➔ Expected backend changes : No change
➔ Test procedure : Reloads to the same page

2. In this test, the searched movie is in the database.
➔ Requirement it is testing : Should display the searched movie and recommended movies and input should not be null
➔ Use case it is testing : Search movie on Movie search engine on Home page.
➔ Inputs to the test : “Toy Story”
➔ Expected outputs :
★ Movie name : “Toy Story”
★ View detail button
★ View movie button
★ Add to watchlist button
➔ Expected backend changes : No change

➔ Test procedure : Reloads to same page(i.e. Home page)

3. In this test, the searched movie is not in the database.
   - ➔ Requirement it is testing : Should display the movies which are somewhat similar to the searched movie in terms of spelling and input should not be null.
   - ➔ Use case it is testing : Search movie on Movie search engine on Home page.
   - ➔ Inputs to the test : “The Story”
   - ➔ Expected outputs :
     - ★ Movie name : “Toy Story”
     - ★ View detail button
     - ★ View movie button
     - ★ Add to watchlist button

There is no such movie named “The Story” so it will display the movie “Toy Story” as a searched movie is similar to “Toy Story” in terms of spelling.

   - ➔ Expected backend changes : No change
   - ➔ Test procedure : Reloads to the same page(i.e. Home page)

4. In this test, the searched cast is in the database.
   - ➔ Requirement it is testing : Should display the movies acted by cast and input should not be null.
   - ➔ Use case it is testing : Search cast on Cast/crew search engine on Home page.
   - ➔ Inputs to the test : “Sandra Oh”
   - ➔ Expected outputs :
     - ★ Movie name : “Double Happiness”
     - ★ View detail button for each movie
     - ★ View movie button for each movie
     - ★ Add to watchlist button for each movie
   - ➔ Expected backend changes : No change
   - ➔ Test procedure : Reloads to the same page

5. In this test, the searched cast is not in the database.
   - ➔ Requirement it is testing : It will display no such actor/actress found and input should not be null
   - ➔ Use case it is testing : Search cast on Cast/crew search engine on

Home page.

- ➔ Inputs to the test : “Smith”
- ➔ Expected outputs : No such actor
- ➔ Expected backend changes : No change
- ➔ Test procedure : Reloads to the same page

6. In this test, the searched crew is in the database.
   - ➔ Requirement it is testing : Should display the movies directed by crew and input should not be null
   - ➔ Use case it is testing : Search crew on Cast/crew search engine on Home page.
   - ➔ Inputs to the test : “Antonia Bird”
   - ➔ Expected outputs :
     - ★ Movie name : “Priest”, “Mad Love”
     - ★ View detail button for each movie
     - ★ View movie button for each movie
     - ★ Add to watchlist button for each movie
   - ➔ Expected backend changes : No change
   - ➔ Test procedure : Reloads to the same page

7. In this test, the searched crew is not in the database.
   - ➔ Requirement it is testing : Should not display any movie since there is no such crew(director) in the database and input should not be null
   - ➔ Use case it is testing : Search crew on Cast/crew search engine on Home page.
   - ➔ Inputs to the test : “James”
   - ➔ Expected outputs : No such director
   - ➔ Expected backend changes : No change
   - ➔ Test procedure : Reloads to the same page

8. In this test, the searched genre is in the database.
   - ➔ Requirement it is testing : Should display the movies of searched genre type and input should not be null
   - ➔ Use case it is testing : Search genre on Genre search engine on Home page.
   - ➔ Inputs to the test : “Comedy”
   - ➔ Expected outputs :
     - ★ Movie name : “Toy Story”, “Four Rooms”, “Money Train” and etc.

- ★ View detail button for each movie
- ★ View movie button for each movie
- ★ Add to watchlist button for each movie

➔ Expected backend changes : No change
➔ Test procedure : Reloads to the same page

9. In this test, the searched genre is not in the database.
   ➔ Requirement it is testing : Should not display any movie as there are no movies of searched genre type in our Database and input should not be null
   ➔ Use case it is testing : Search genre on Genre search engine on Home page.
   ➔ Inputs to the test : “Emotion”
   ➔ Expected outputs : No such genre
   ➔ Expected backend changes : No change
   ➔ Test procedure : Reloads to the same page

10.
   ➔ Requirement it is testing : Should log out the user and redirect to login page
   ➔ Use case it is testing : Click on Logout button on Home page
   ➔ Inputs to the test : No input
   ➔ Expected outputs : No output
   ➔ Expected backend changes : No change
   ➔ Test procedure : Displays login page

11.
   ➔ Requirement it is testing : Should show details of a movie
   ➔ Use case it is testing : Click on View detail button for a movie
   ➔ Inputs to the test : For example click on view detail button for “Double Happiness” movie
   ➔ Expected outputs :
      - ★ Movie name: : “Double Happiness”
      - ★ Lead Roles: : “Sandra Oh”
      - ★ Languages available : English, 广州话 / 廣州話
      - ★ Ratings: : 5.5(out 0f 10)
      - ★ R Rated : false
      - ★ Duration : 87.0 (in mins)

   ➔ Expected backend changes : No change

- ➔ Test procedure : Redirects to a page having details like Title, Lead_roles, Languages available, Ratings, R Rated and Duration of the movie.

12. In this test, the movie is not added in watching_now page for the user
    - ➔ Requirement it is testing : Should add the movie in Watching_now page.
    - ➔ Use case it is testing : Click on View movie button for a movie
    - ➔ Inputs to the test : For example click on view movie button for “Double Happiness” movie
    - ➔ Expected outputs : No output
    - ➔ Expected backend changes : Adds a relationship from user to movie in watching_now relationship
    - ➔ Test procedure : Reloads to the same page

13. In this test, the movie is already added in watching_now page for the user
    - ➔ Requirement it is testing : View movie
    - ➔ Use case it is testing : Click on View movie button for a movie
    - ➔ Inputs to the test : For example click on view movie button for “Double Happiness” movie
    - ➔ Expected outputs : No output
    - ➔ Expected backend changes : No change since there is relationship between the user and movie already
    - ➔ Test procedure : Reloads to the same page

14. In this test, the movie is not added in watchlist for the user
    - ➔ Requirement it is testing : Should add movie to Add to watchlist page
    - ➔ Use case it is testing : Add to watchlist button for a movie
    - ➔ Inputs to the test : Click the Add to watchlist button for a movie, for example click on Add to watchlist for “Double Happiness” movie
    - ➔ Expected outputs : No output
    - ➔ Expected backend changes : Adds a relationship from user to movie in watchlist relationship
    - ➔ Test procedure : Reloads to the same page(i.e. Home page)

15. In this test, the movie is added in watchlist for the user
    - ➔ Requirement it is testing : Add to watchlist
    - ➔ Use case it is testing : Click on Add to watchlist button for a movie

- ➔ Inputs to the test : For example click on Add to watchlist button of “Double Happiness” movie
- ➔ Expected outputs : No output
- ➔ Expected backend changes : No change since the movie is already added in watchlist for the user
- ➔ Test procedure : Reloads to the same page(i.e. Home page)

In WatchList Page:

1.
   - ➔ Requirement it is testing : Should display all movies added to watchlist by clicking the “Add to watchlist“ button on any page containing this button.
   - ➔ Use case it is testing : Click on WatchList button
   - ➔ Inputs to the test : No input
   - ➔ Expected outputs : “Double Happiness” and “Jumanji” are added in Watchlist page
     - ★ Movie name: : “Double Happiness”
     - ★ View details button
     - ★ View movie
     - ★ Remove from Watchlist button

     - ★ Movie name: : “Jumanji”
     - ★ View details button
     - ★ View movie
     - ★ Remove from Watchlist button

   - ➔ Expected backend changes : No change
   - ➔ Test procedure : Reloads to the same page

2.
   - ➔ Requirement it is testing : Should display details of a movie
   - ➔ Use case it is testing : Click on View detail button for a movie
   - ➔ Inputs to the test : For example click on view detail button for “Double Happiness” movie
   - ➔ Expected outputs :
     - ★ Movie name: : “Double Happiness”

- ★ Lead Roles: : “Sandra Oh”
- ★ Languages available : English, 广州话 / 廣州話
- ★ Ratings: : 5.5(out 0f 10)
- ★ R Rated : false
- ★ Duration : 87.0 (in mins)

➔ Expected backend changes : No change
➔ Test procedure : A page having details like Title, Lead_roles, Languages available, Ratings, R Rated and Duration of the movie.

3. In this test, the movie is not added in watching_now page for the user
   ➔ Requirement it is testing : Should add the movie to Watching_now page
   ➔ Use case it is testing : Click on View movie button for a movie
   ➔ Inputs to the test : For example click on view movie button for “Double Happiness” movie
   ➔ Expected outputs : No output
   ➔ Expected backend changes : Adds a relationship from user to movie in watching_now relationship
   ➔ Test procedure : Reloads to the same page

4. In this test, the movie is added in watching_now relation for the user
   ➔ Requirement it is testing : View movie
   ➔ Use case it is testing : Click on View movie button for a movie
   ➔ Inputs to the test : For example click on view movie button for “Double Happiness” movie
   ➔ Expected outputs : No output
   ➔ Expected backend changes : No change since there is relationship between the user and movie already
   ➔ Test procedure : Reloads to the same page

5.
   ➔ Requirement it is testing : Should remove the movie from watchlist
   ➔ Use case it is testing : Click on Remove from watchlist for a movie
   ➔ Inputs to the test : For example click on remove from watchlist for “Double Happiness” movie
   ➔ Expected outputs : No output
   ➔ Expected backend changes : Removes the relationship from user to

Double Happiness in WatchList relationship

➔ Test procedure : Reloads to the same page

In Watching_now Page:

1.

➔ Requirement it is testing : Should display all movies added to watching_now by clicking the “View movie“ button on any page containing this button.

➔ Use case it is testing : Click on Watching_now button

➔ Inputs to the test : No input

➔ Expected outputs : “Double Happiness” and “Jumanji” are added in Watchlist page

★ Movie name: : “Double Happiness”

★ View details button

★ Remove from Watching_now button

★ Rate Movie

★ Movie name: : “Jumanji”

★ View details button

★ Remove from Watching_now button

★ Rate Movie

➔ Expected backend changes : No change

➔ Test procedure : Reloads to the same page

2.

➔ Requirement it is testing : Should display details of a movie

➔ Use case it is testing : Click on View detail button for a movie

➔ Inputs to the test : For example click on view detail button for “Double Happiness” movie

➔ Expected outputs :

★ Movie name: : “Double Happiness”

★ Lead Roles: : “Sandra Oh”

★ Languages available : English, 广州话 / 廣州話

★ Ratings: : 5.5(out 0f 10)

★ R Rated : false

★ Duration : 87.0 (in mins)

- ➔ Expected backend changes : No change
- ➔ Test procedure : A page having details like Title, Lead_roles, Languages available, Ratings, R Rated and Duration of the movie.

3.

- ➔ Requirement it is testing : Should remove the movie from Watching_now page.
- ➔ Use case it is testing : Remove movie from watching_now button
- ➔ Inputs to the test : “Toy Story”
- ➔ Expected outputs : No output
- ➔ Expected backend changes : Removes the Watching_now relationship from user to “Toy Story”
- ➔ Test procedure : Reloads to the same page

4. In this test, the user has never rated the movie.

- ➔ Requirement it is testing : Should be done rating of movie by user and after rating click on ok and if user don’t want to rate even if user clicked on the button then the user can opt cancel option.
- ➔ Use case it is testing : Click on Rate movie button
- ➔ Inputs to the test : For ex: “Toy Story” and user will need to give point_based rating or like/dislike
- ➔ Expected outputs : No output
- ➔ Expected backend changes : Add the relationship Rated from user to movie with attributes point_based_rating or opt for like/dislike given by the user.
- ➔ Test procedure : A pop up will pop having the option to like or dislike and give rating out of 10(range 1-10) and after giving rating, redirect to the Watching_now page.

5. In this test, the user has rated the movie in the past.

- ➔ Requirement it is testing : Should be done rating of movie by user and after rating click on ok and if user don’t want to rate even if user clicked on the button then the user can opt cancel option.
- ➔ Use case it is testing : Click on Rate movie button

- ➔ Inputs to the test : For ex: “Toy Story” and user will need to give point_based rating or opt for like/dislike
- ➔ Expected outputs : No output
- ➔ Expected backend changes : Update the attributes point_based_rating and like/dislike the value given by the user.
- ➔ Test procedure : A pop up will pop having the option to like or dislike and give rating out of 10(range 1-10) and after giving rating, redirect to the Watching_now page.

In Friends Page:

1. In this test, user has friends with usernames such as “Nobita”,”Doremon”,”Shizuka”,”Suneo”
   - ➔ Requirement it is testing : On clicking the send request button a friend request to the user whose button is clicked
   - ➔ Use case : *Send Request*
   - ➔ Inputs to the test : Username to whom request is being sent
     - ◆ Example Input:
       - Current user
       - User: “Nobita”(To whom the request is to be sent)
   - ➔ Expected outputs : No output
   - ➔ Expected backend changes : A request will be added to the table friend request
   - ➔ Redirection : Redirected to the friends page
2. Requirement it is testing : On clicking the Recommend button a list of the current user’s watched list will appear
   - ➔ Use case : *Recommend*
   - ➔ Inputs to the test : Username to whom request is being sent
     - ◆ Example Input:
       - Current user
       - User: “Nobita”(Whose button the user clicks)
   - ➔ Expected outputs : List of movies the current user has watched
   - ➔ Expected backend changes : No changes to the backend yet
   - ➔ Redirection : Redirected to the recommend page

In Friend Request Page:

1. Requirement it is testing : On clicking the cancel button rejects that friend request whose button is clicked
    - ➔ Use case : *Cancel Request*
    - ➔ Inputs to the test : username whose request is being cancelled
    - ➔ Expected backend changes : Remove the request from the list of friend requests table
        - ◆ Example Input:
            - User: “Nobita”(whose request needs to be deleted)
    - ➔ Expected output : No output
    - ➔ Redirection : Redirected to the friend requests page
2. Requirement it is testing : On clicking the Accept request button the user which sent the request can now follow the current user
    - ➔ Use case : *Accept Request*
    - ➔ Inputs to the test : Username of person who sent the request
        - ◆ Example Input:
            - Current user
            - User: “Nobita”(To whom the request is to be sent)
    - ➔ Expected outputs : No output
    - ➔ Expected backend changes : A new entry is added to the friends table
    - ➔ Redirection : Redirected to the friends page

In Recommend Page:

1. List of the current user’s watched movies appear here and every movie has a button recommend. Example movies:- “Copycat”,”Othelle” etc.
    - ➔ Requirement : On clicking the Recommend button a recommendation will be sent to the user for which we recommend it. At Least one movie should have been watched by the current user.
    - ➔ Use case : *Recommend movie*
    - ➔ Inputs to the test : Username to whom recommend is being sent and the movie name
        - ◆ Example Input:
            - Current user
            - User: “Nobita” (to whom we need to recommend)
            - Movie: “Copycat”

- ➔ Expected outputs : No output
- ➔ Expected backend changes : A recommendation will be added to the table with the current user and user to which the recommendation is being done and the movie name
- ➔ Redirection : Redirected to the recommend page

In Movies Recommended by friends Page:

1. In this test, user has friends with usernames "Nobita","Doremon","Shizuka","Suneo"
   - ➔ Requirement it is testing : Displaying the movies recommended by the user's friends.
   - ➔ Use case it is testing : Movies recommended by friends page
   - ➔ Inputs to the test : No input
   - ➔ Expected outputs :
     - ❖
       - ★ Movie name : "Jumanji"
       - ★ View Details :
         - ➢ Language : English
         - ➢ R-Rated : PG-13
         - ➢ Rating : 8.5
         - ➢ Duration : 1hr 59mins
         - ➢ Genre : Comedy,Adventure,Fantasy
       - ★ Recommended by : "Doremon"
       - ★ Add to Watchlist Button
       - ★ View movie Button
     - ❖
       - ★ Movie name : "Twilight"
       - ★ View Details :
         - ➢ Language : English
         - ➢ R-Rated : PG-13
         - ➢ Rating : 7.6
         - ➢ Duration : 2hr 10mins
         - ➢ Genre : Romance,Melodrama,Fantasy
       - ★ Recommended by : "Shizuka"
       - ★ Add to Watchlist Button
       - ★ View movie Button

- ❖
  - ★ Movie name : "Ghazi"
  - ★ View Details :
    - ➢ Language : Telugu
    - ➢ R-Rated : PG-13
    - ➢ Rating : 7.2
    - ➢ Duration : 2hr 3mins
    - ➢ Genre : War,History,Drama
  - ★ Recommended by : ”Shizuka”
  - ★ Add to Watchlist Button
  - ★ View movie Button
- ➔ Expected backend changes : No change
- ➔ Test procedure : Input is not needed and output is displayed on this page.

In Profile Page: (Edit Access is given by the Admin and cannot be edited by user/Editor)

1. In this test, username is Jk1997
   - ➔ Requirement it is testing : Displaying the profile of user
   - ➔ Use case it is testing : Profile page
   - ➔ Inputs to the test : No input
   - ➔ Expected outputs :
     - ❖
       - ★ Name : "Jeon Jungkook"
       - ★ Username : Jk1997
       - ★ Date of Birth : 01-08-1997
       - ★ Gender : Male
       - ★ Language : Korean
       - ★ Edit_Access : No
   - ➔ Expected backend changes : No change
   - ➔ Test procedure : Input is not needed and output is displayed on this page.
2. In this test, user is updating the name
   - ➔ Requirement it is testing : Editing the user details and character the limit is 30.
   - ➔ Use case it is testing : Edit button for each detail
   - ➔ Inputs to the test :
     - ❖

★ Name : "Jeon Jungkookie"

➔ Expected outputs :

❖

★ Name : "Jeon Jungkookie"
★ Username : Jk1997
★ Date of Birth : 01-08-1997
★ Gender : Male
★ Language : Korean
★ Edit_Access : No

➔

➔ Expected backend changes : Updates the value of property 'Name' in "Jk1997" node to "Jeon Jungkookie"

➔ Test procedure : Input is given by typing the name. Reloads on the same page .

3. In this test, user is updating the date of birth
   ➔ Requirement it is testing : Editing the user details
   ➔ Use case it is testing : Edit button for each detail
   ➔ Inputs to the test :
   ★ Date of Birth : 02-08-1997
   ➔ Expected outputs :
   ★ Name : "Jeon Jungkookie"
   ★ Username : Jk1997
   ★ Date of Birth : 02-08-1997
   ★ Gender : Male
   ★ Language : Korean
   ★ Edit_Access : No
   ➔ Expected backend changes : Updates the value of property 'Dob' in "Jk1997" node to 02-08-1997
   ➔ Test procedure : Input is given by typing the name. Reloads on the same page .
4. In this test, user is updating the gender
   ➔ Requirement it is testing : Editing the user details and Gender is selected by user from the options mentioned in the page
   ➔ Use case it is testing : Edit button for each detail
   ➔ Inputs to the test :
   ★ Gender : Female
   ➔ Expected outputs :
   ★ Name : "Jeon Jungkookie"

- ★ Username : Jk1997
- ★ Date of Birth : 02-08-1997
- ★ Gender : Female
- ★ Language : Korean
- ★ Edit_Access : No

➔ Expected backend changes : Updates the value of property "Gender' in "Jk1997" node to female

➔ Test procedure : Input is given by selecting the gender. Reloads on the same page .

5. In this test, user is updating the language

➔ Requirement it is testing : Editing the user details and Language is selected by user from the options mentioned in the page

➔ Use case it is testing : Edit button for each detail

➔ Inputs to the test :

- ★ Language : English

➔ Expected outputs :

- ★ Name : "Jeon Jungkookie"
- ★ Username : Jk1997
- ★ Date of Birth : 02-08-1997
- ★ Gender : Female
- ★ Language : English
- ★ Edit_Access : No

➔ Expected backend changes : Updates the value of property "Language' in "Jk1997" node to English

➔ Test procedure : Input is given by selecting the language. Reloads on the same page .

Conclusion:

Pages:

Home page,Registration page,Login page,Friends page,Profile page,movies recommended by friends page,WatchList,View Details,Content based and demographic based Filtering are done.

Favourites page,admin page - not done

Reason- Due to insufficient amount of time and due to lack of time management and absence of a team member due to covid situation we were unable to complete what we initially expected.

Use cases:

Add movie/actor/actress by editor is not done.

Github link:

https://github.com/sindhu-2507/387-FINAL-SUBMISSION/tree/main/387-FINAL-SUBMISSION