Project Title: MultimediaHub

Group Members:

- Mustafa İzzet Yumuşak (2020510075)
- Eren Çağlar Erdoğan (2019510034)
- Ömer Faruk KESKİN (2018510145)

Abstract:

MultimediaHub, a dynamic web application for multimedia enthusiasts, is implemented using React for the frontend and Node.js for the backend. This technical manual provides a detailed overview of the application's architecture, components, completion status, and considerations for future development.

Completion Report:

In the developmental journey of the MultimediaHub web application, several noteworthy milestones have been achieved, laying the groundwork for a vibrant platform catering to multimedia enthusiasts. The integration of React for the frontend and Node.js for the backend stands out as a major success, providing users with a robust foundation for exploring, sharing, and interacting with multimedia content. While core functionalities have been effectively implemented, the project did encounter certain challenges, which were tactfully addressed during the development process.

One particular triumph is the successful establishment of a comprehensive database system, leveraging technologies such as MySQL/MsSQL. This milestone ensures the efficient storage and retrieval of multimedia data, user information, and other relevant content. The integration of backend logic for user registration and profile creation adds an extra layer of security and personalization to the user experience.

However, the developmental path did present challenges, notably in the intricacies of frontend design complexities. The React components, crucial for shaping the user interface, demanded meticulous attention and iterative improvements to achieve an optimal user experience. Collaborative problem-solving sessions played a pivotal role in overcoming these challenges, resulting in a visually appealing and interactive frontend.

The project timeline underwent slight adjustments to accommodate challenges in frontend development, ensuring that a comprehensive testing and polishing phase remains integral to the project. The revised plan aims to finalize React frontend development, conduct thorough testing and optimization, and adequately prepare for the upcoming demonstration.

Despite these challenges, the team's unwavering commitment to delivering a seamless user experience has prevailed. The successful integration of React components and Node.js functionality serves as a testament to the team's dedication in achieving the project's goals. The implementation, adhering to the client-server architecture, is complemented by a clear functional decomposition of the application's primary components.

In summary, the completion report for MultimediaHub emphasizes the successful implementation of core functionalities, robust database integration, and the adept resolution of challenges encountered during development. The team's adaptability, problem-solving skills, and commitment to excellence continue to propel the project forward. The next phases will concentrate on testing, optimization, and the final demonstration, ensuring a polished and user-friendly multimedia platform

Functional Decomposition:

In the implementation of MultimediaHub, we have adopted a client-server architecture using Node.js. While this architecture doesn't strictly adhere to the Model-View-Controller (MVC) pattern, we can break down the functional components into distinct client-side and server-side responsibilities.

Client-Side Functional Decomposition:

React Components:

Overview: React components represent the views in our client-server architecture.

Purpose: Responsible for rendering the user interface and managing user interactions on the client-side.

Components:

Home: Displays the latest posts.

Explore: Allows users to browse and search for multimedia content.

UserProfile: Presents user-specific information, including reviews, lists, and diary entries.

ReviewForm: Enables users to submit reviews for films, music, and games.

ListForm: Facilitates the creation and management of user-specific lists.

Diary: Allows users to log and reflect on their multimedia viewing experiences.

Showcase: Provides a personalized space for users to showcase their favorite multimedia items.

Server-Side Functional Decomposition (Node.js):

Express Server:

Overview: Serves as the backend server for handling client requests and managing data.

Purpose: Responsible for routing, request handling, and communication with the database.

Components:

Routes: Define endpoints for handling various client requests (e.g., user authentication, multimedia data retrieval).

Middleware: Executes functions between the request and the final route handler.

Database (MySQL):

Overview: Stores and retrieves multimedia data, user information, and application-related content.

Purpose: Manages the persistent data needed for the application's functionality.

Components:

Users Table: Stores user information (e.g., username, email, profile details).

Multimedia Table: Stores information about films, music, and games (e.g., title, reviews, ratings).

Lists Table: Stores user-created lists.

Diary Table: Stores user-recorded viewing experiences.

Authentication Module:

Overview: Manages user authentication and authorization.

Purpose: Ensures secure access to user-specific functionalities and data.

Components:

Login: Verifies user credentials and generates authentication tokens.

Register: Adds new users to the database with secure password storage.

Middleware: Validates and authorizes user requests based on authentication tokens.

Interaction Flow:

User Interaction:

Users interact with the React components on the client-side, initiating requests.

React components make HTTP requests to specific endpoints on the Express server.

Server Processing:

Express server routes process incoming requests, interact with the database, and execute necessary logic.

Authentication middleware ensures secure access to user-specific functionalities.

Database Interaction:

The server communicates with the database to retrieve or store relevant data.

Response to Client:

The server sends a response back to the React components on the client-side.

React components update the user interface based on the server response.

This functional decomposition outlines the key components and their responsibilities in our client-server architecture using Node.js, providing a clear understanding of how the application's primary functions are distributed between the client and server.

High-level Organization:

Home Page:

Latest posts

CreatepostPage:

Films

Music

Games

Media Search

User Profile:

Lists

Showcase

Media Detail Page:

Write a Review

İnformation about media

Users:

User search

following users

Login Page:

Username/Email field

Password field

Login button

Option to register if not already a user

Register Page:

Username

Email

Password

Confirm Password

Register button

Link to the login page for existing users

Clickstreams:

User Publishes a Review:

I. User navigates to the "Media Detail Section" section.

II. Completes the required fields for the review, including title, content, and rating.

III. Submits the review for publication.

IV. System processes the review, updates the database with the new review entry.

User Explore Multimedia:

I. User visits the "Create a post" section, divided into Films, Music, and Games.

II. Filters or searches for specific multimedia based on preferences.

III. Clicks on a multimedia item to view its details.

IV. System retrieves multimedia details from the database and presents them to the user.

User Creates a List:

I. User goes to the "Lists" section.

II. Creates a new list, giving it a title and description.

III. Adds multimedia items to the list.

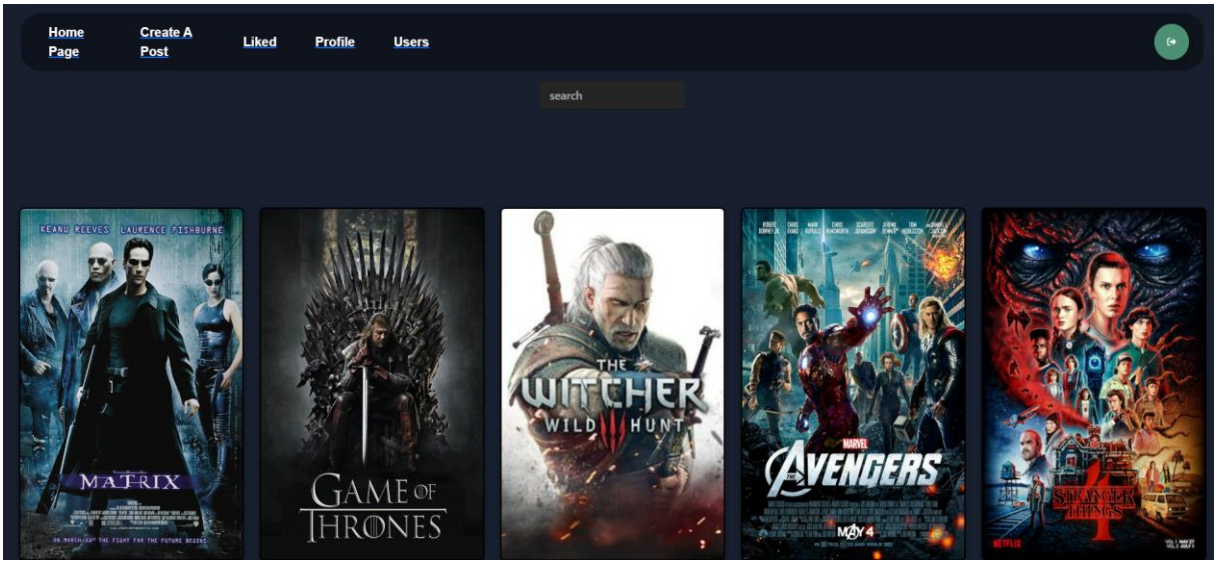
IV. Saves the list for future reference.

V. System updates the database with the new list entry.

Implementation:

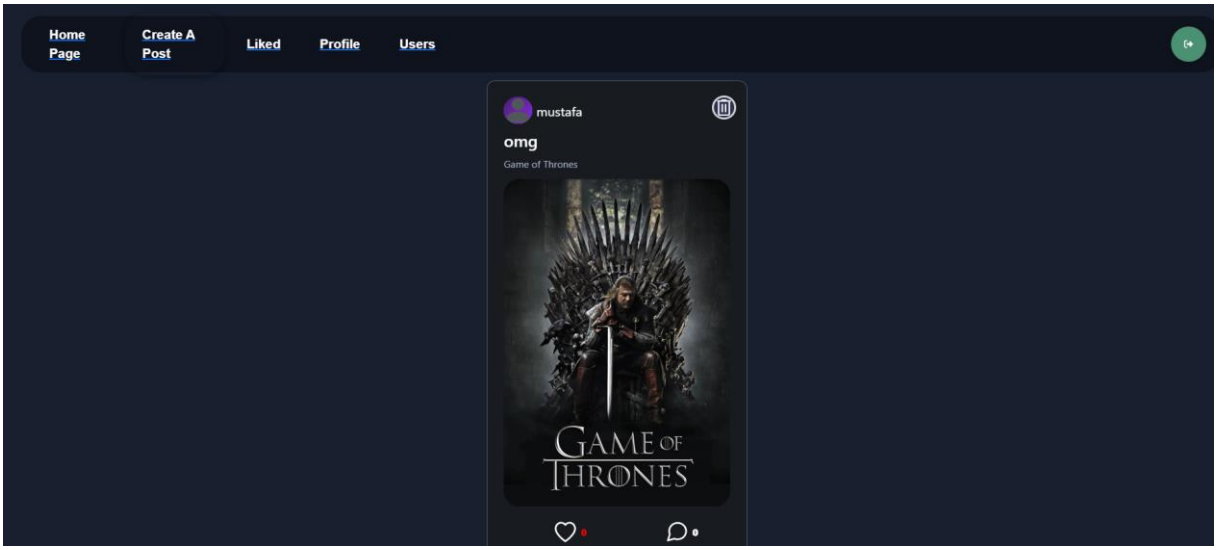
Create A Post Page

Text can be written on this page for creating posts, and it can be shared or added to a list.Medias can be searched by user

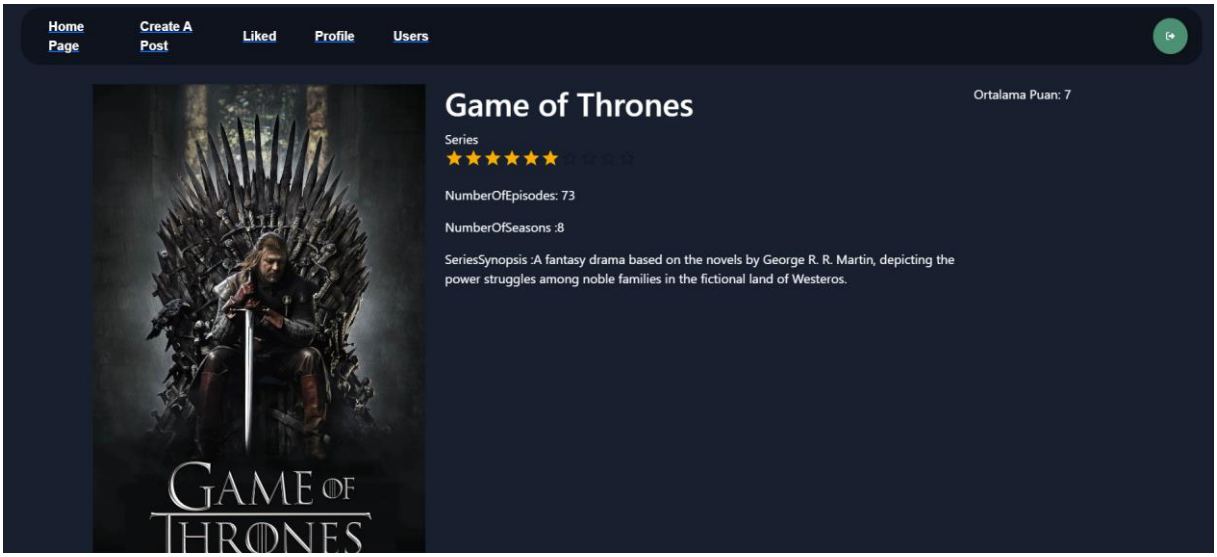


Home Page:

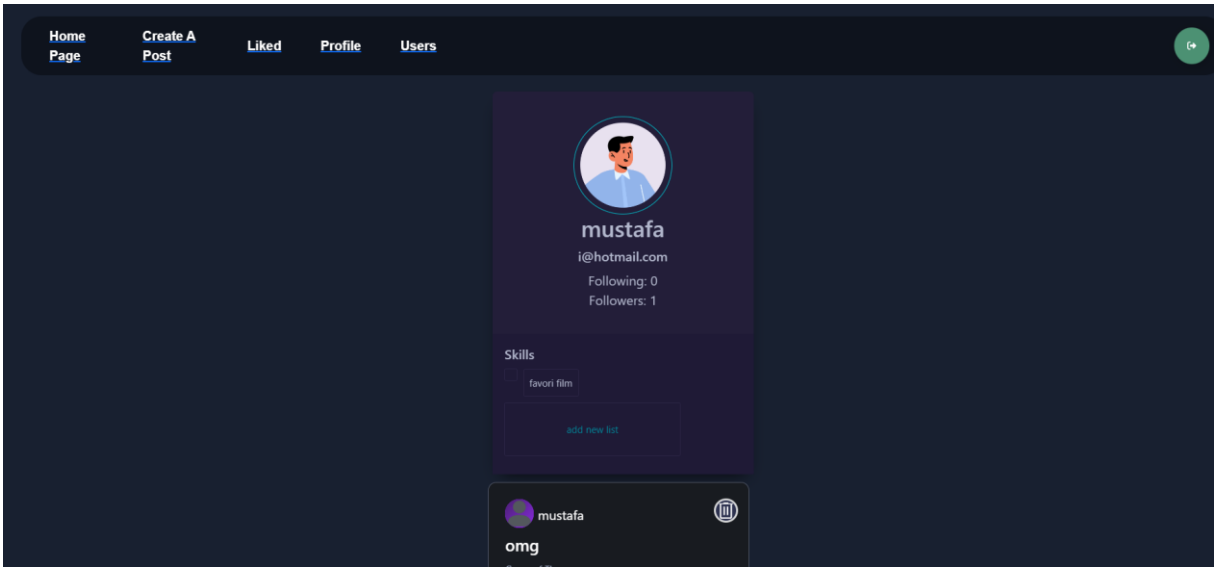
On the home page, posts can be liked and commented on. If a post belongs to us, it can be deleted.



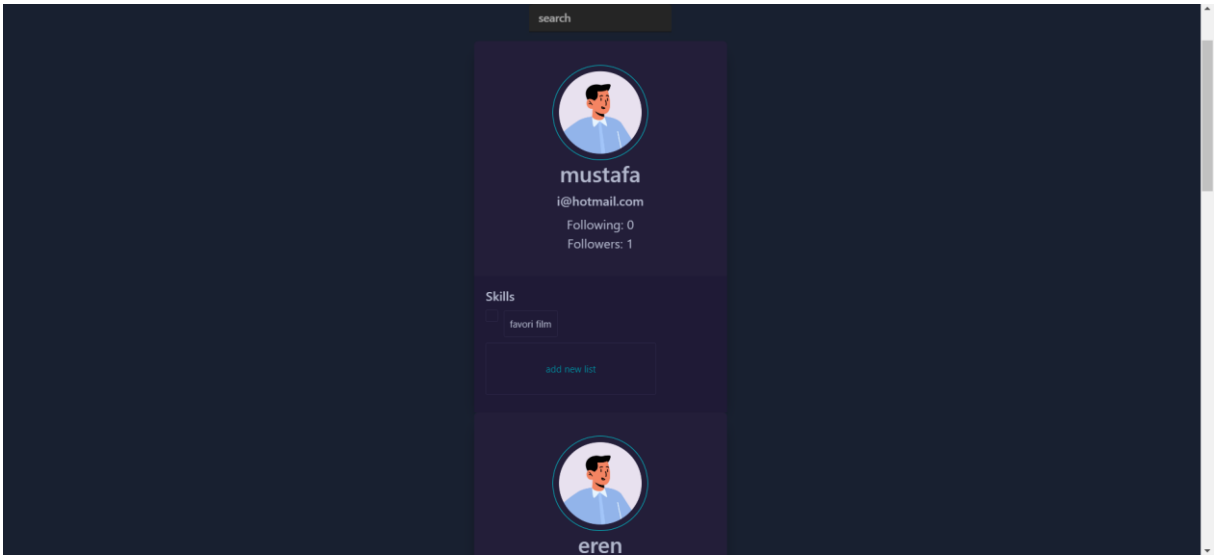
Media Detail Page: When the post is clicked, we are directed to the media details, where we can review and comment on the media.



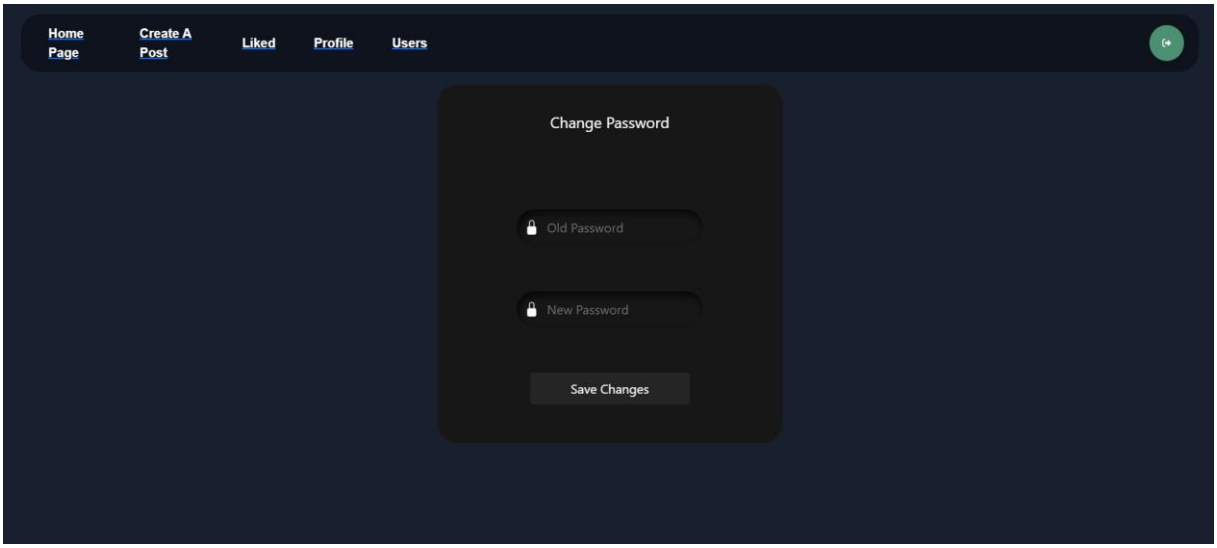
Profile Page: "We can view posts shared by the user and lists that contain media."



User Page : We can view all users, search for them, and follow them.



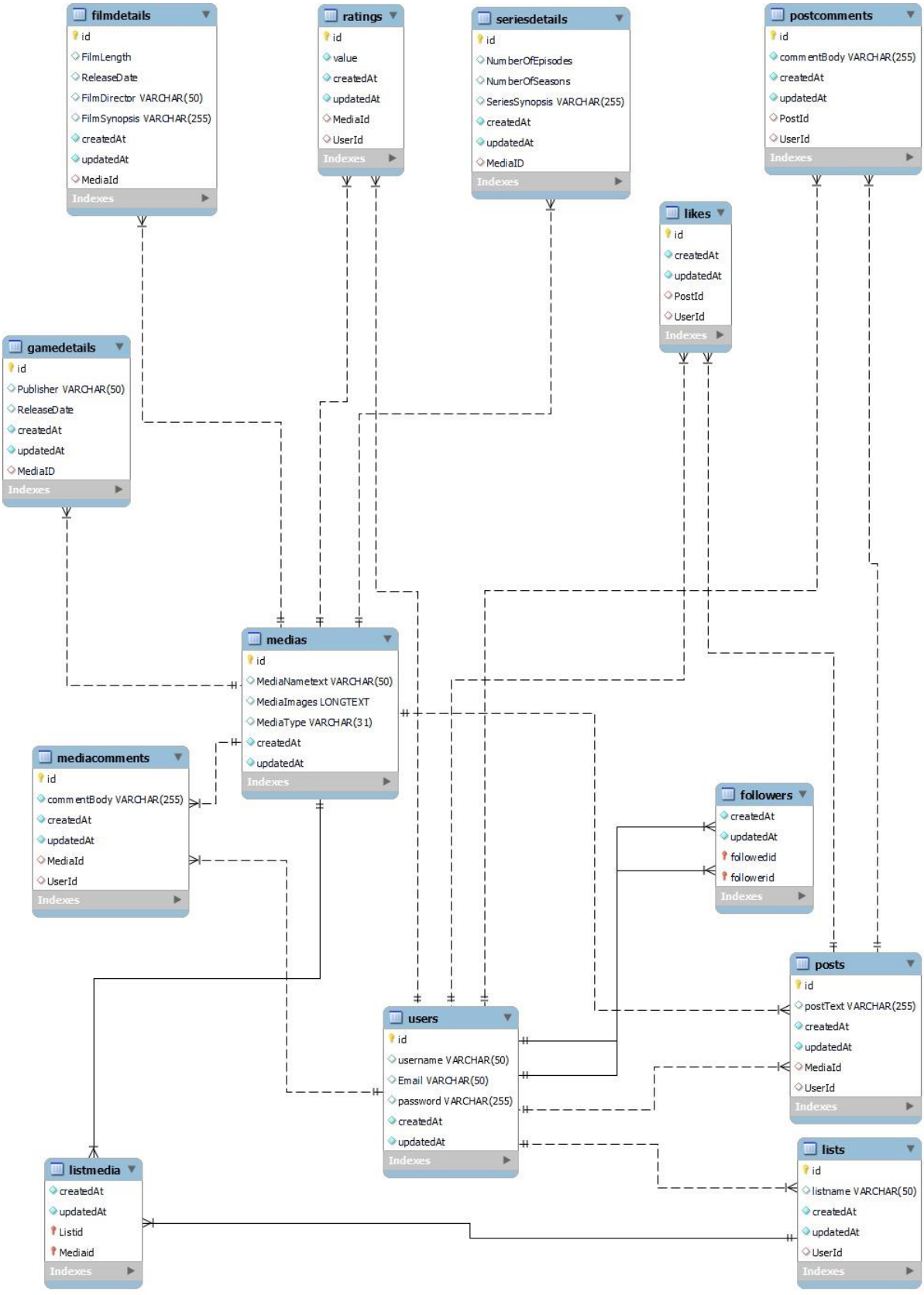
Change Password Page:Password can be changed by user



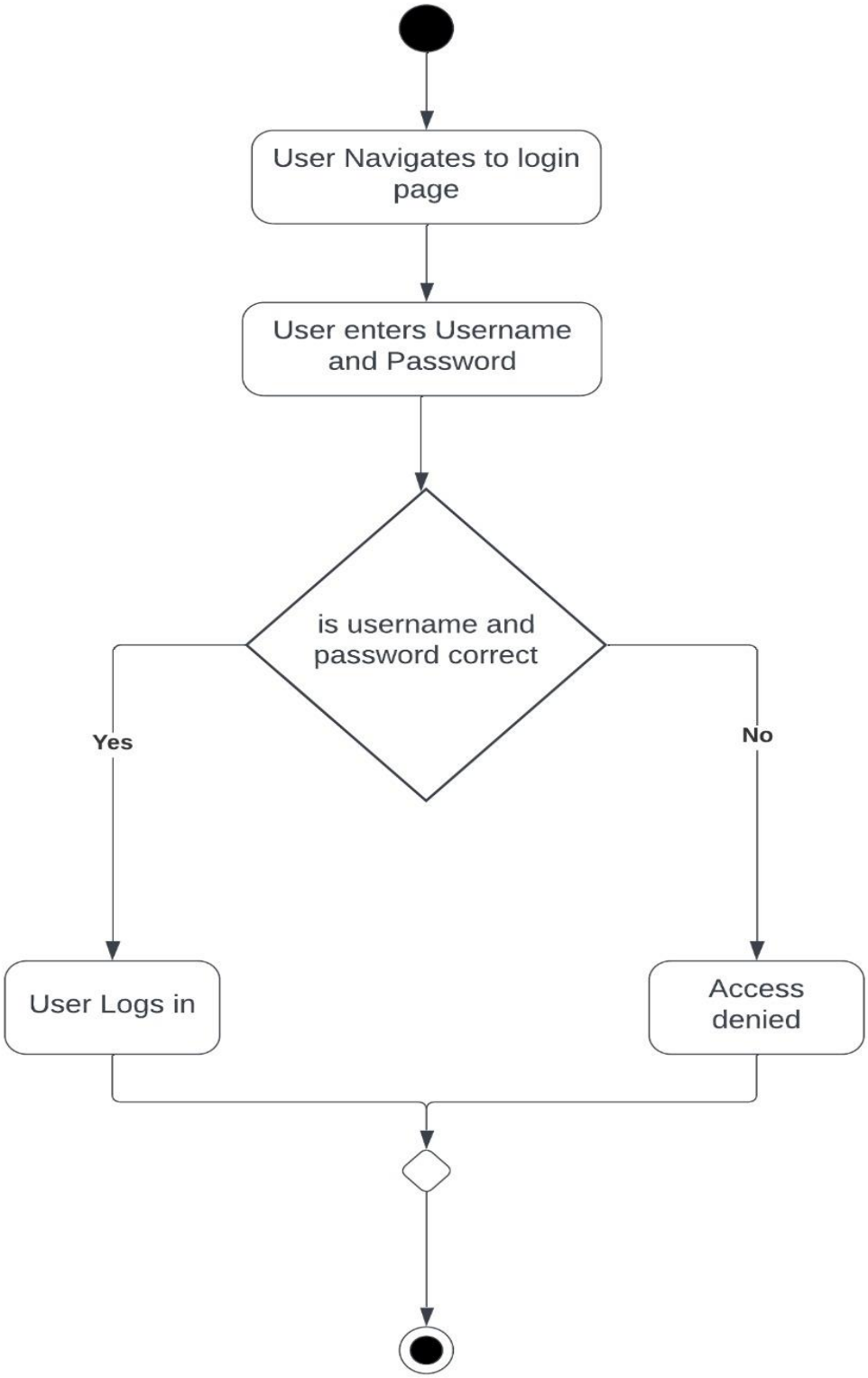
Future Work:

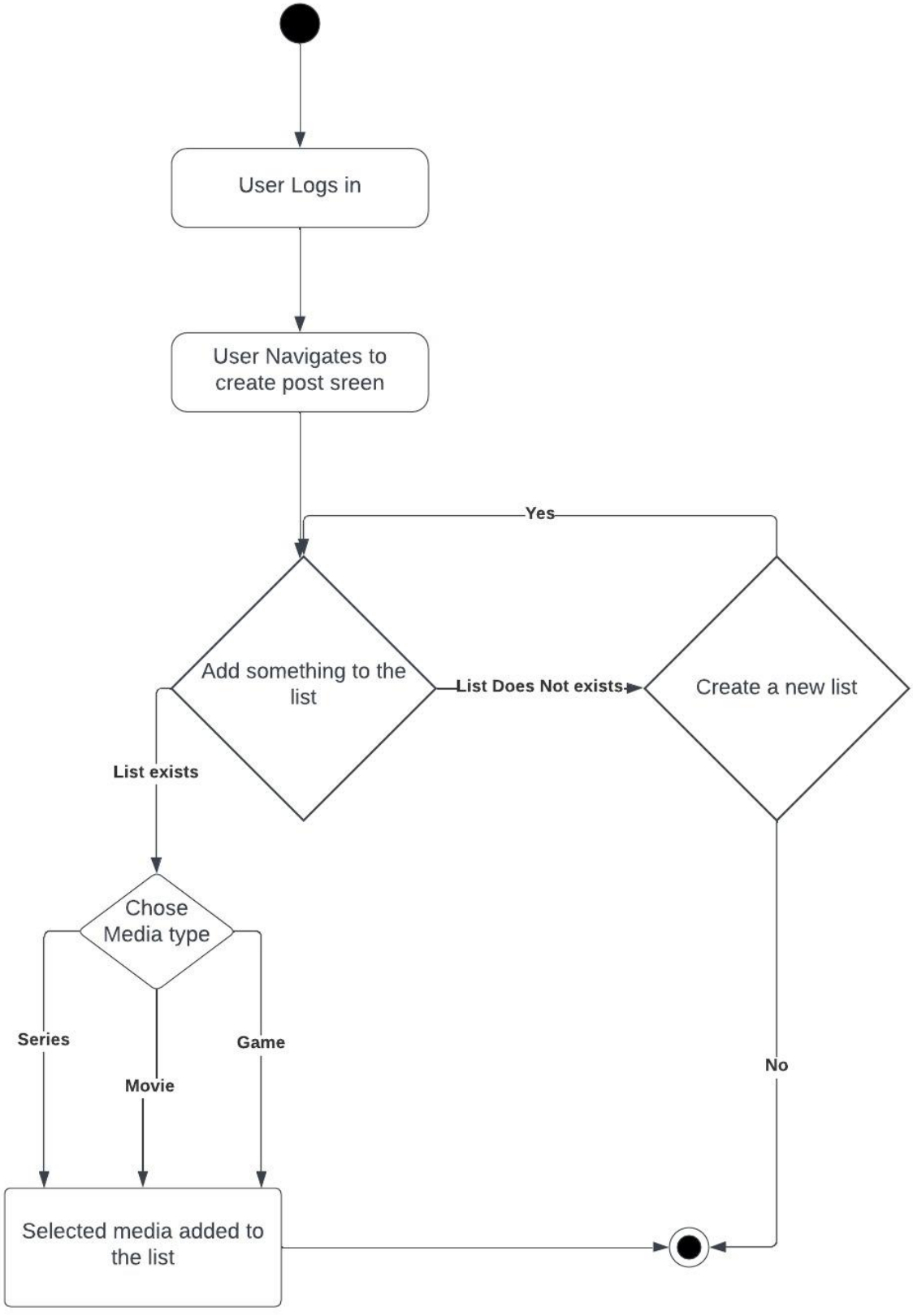
In addition to the considerations mentioned earlier, an essential aspect of future work involves the integration of a real-time chat feature. This feature would allow users to communicate instantly, fostering a more interactive and community-driven environment. The chat functionality could be implemented using WebSocket technology or other suitable technologies that ensure timely and efficient communication.

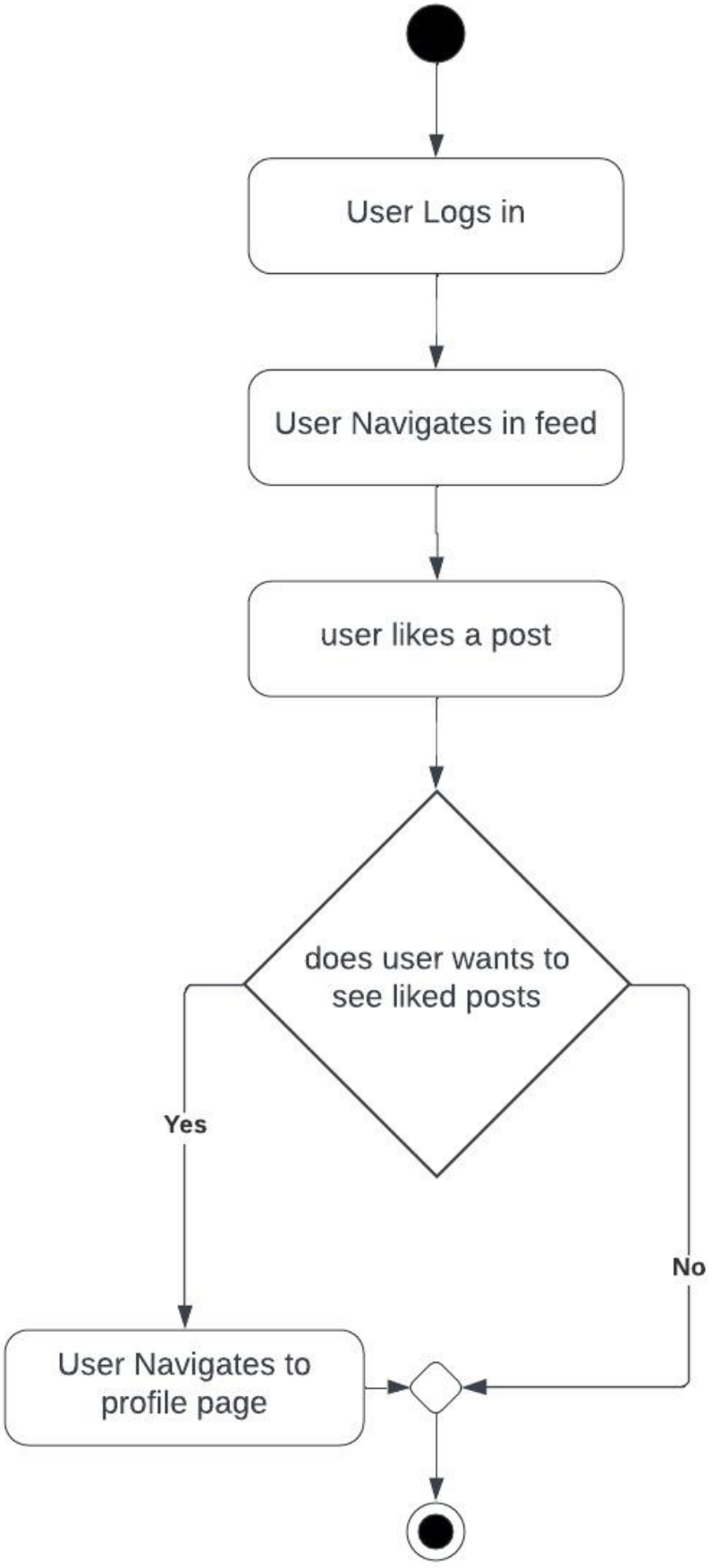
ER DIAGRAM

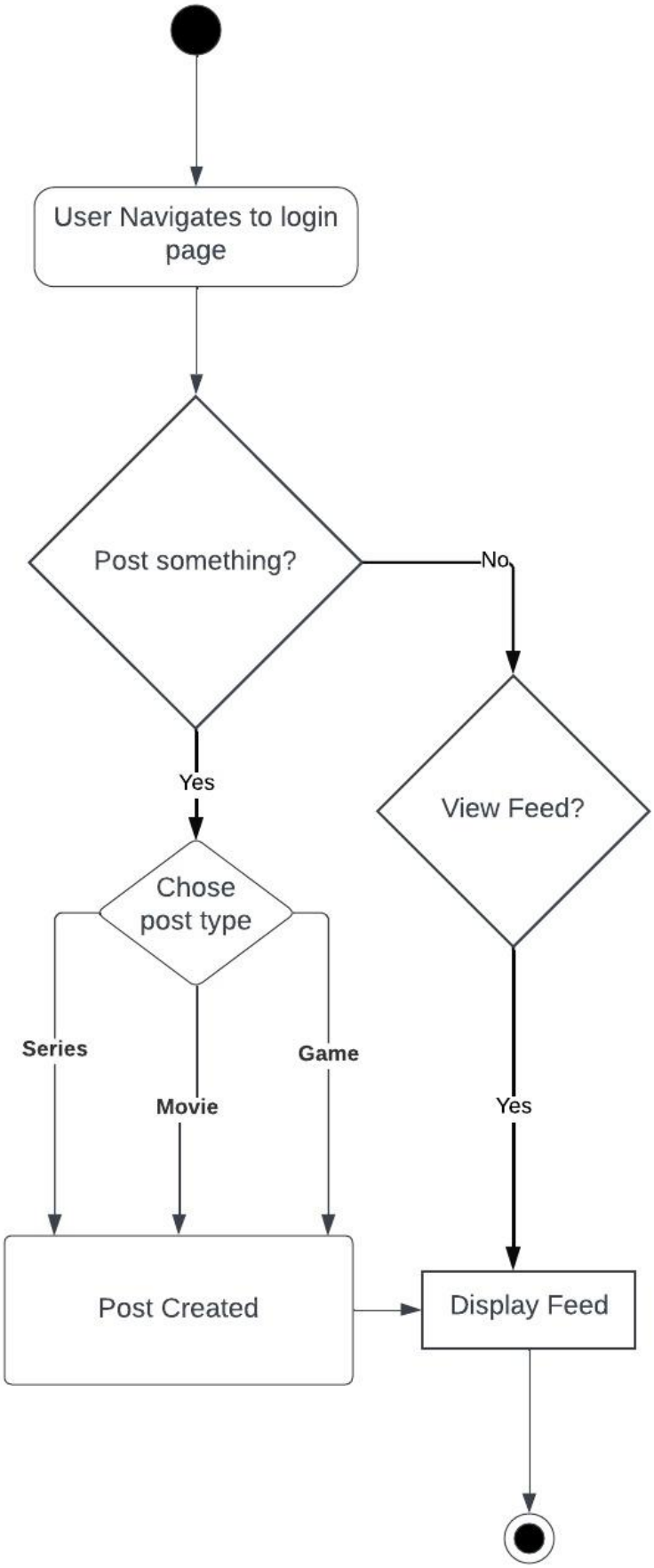

filmdetails
id
FilmLength
ReleaseDate
FilmDirector VARCHAR(50)
FilmSynopsis VARCHAR(255)
createdAt
updatedAt
MediaId
Indexes
ratings
id
value
createdAt
updatedAt
MediaId
UserId
Indexes
seriesdetails
id
NumberOfEpisodes
NumberOfSeasons
SeriesSynopsis VARCHAR(255)
createdAt
updatedAt
MediaID
Indexes
postcomments
id
commentBody VARCHAR(255)
createdAt
updatedAt
PostId
UserId
Indexes
likes
id
createdAt
updatedAt
PostId
UserId
Indexes
gamedetails
id
Publisher VARCHAR(50)
ReleaseDate
createdAt
updatedAt
MediaID
Indexes
medias
id
MediaNametext VARCHAR(50)
MediaImages LONGTEXT
MediaType VARCHAR(31)
createdAt
updatedAt
Indexes
mediacomments
id
commentBody VARCHAR(255)
createdAt
updatedAt
MediaId
UserId
Indexes
followers
createdAt
updatedAt
followedid
followerid
Indexes
posts
id
postText VARCHAR(255)
createdAt
updatedAt
MediaId
UserId
Indexes
users
id
username VARCHAR(50)
Email VARCHAR(50)
password VARCHAR(255)
createdAt
updatedAt
Indexes
listmedia
createdAt
updatedAt
Listid
Mediaid
Indexes
lists
id
listname VARCHAR(50)
createdAt
updatedAt
UserId
Indexes

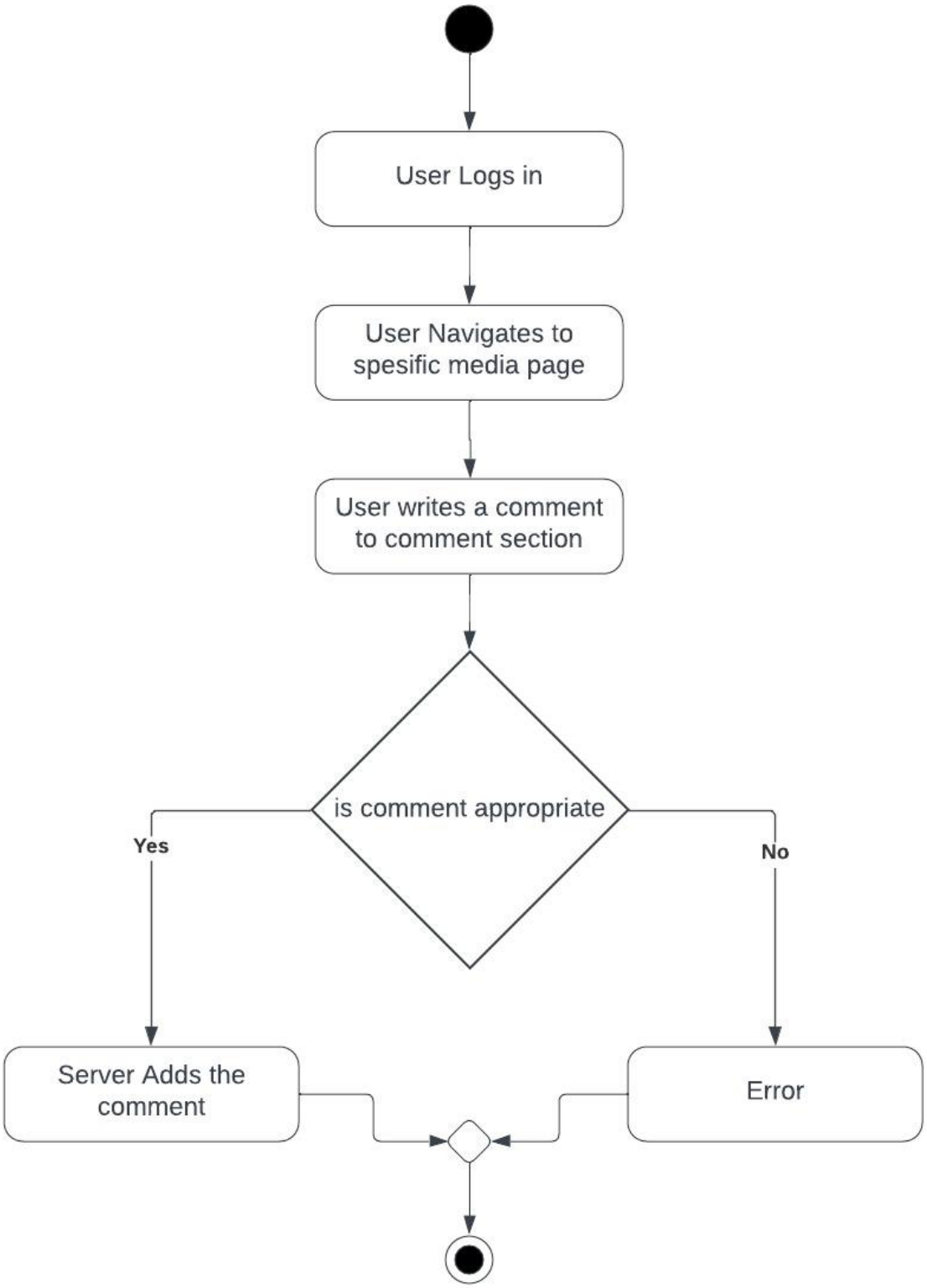
ACTIVITY DIAGRAMS



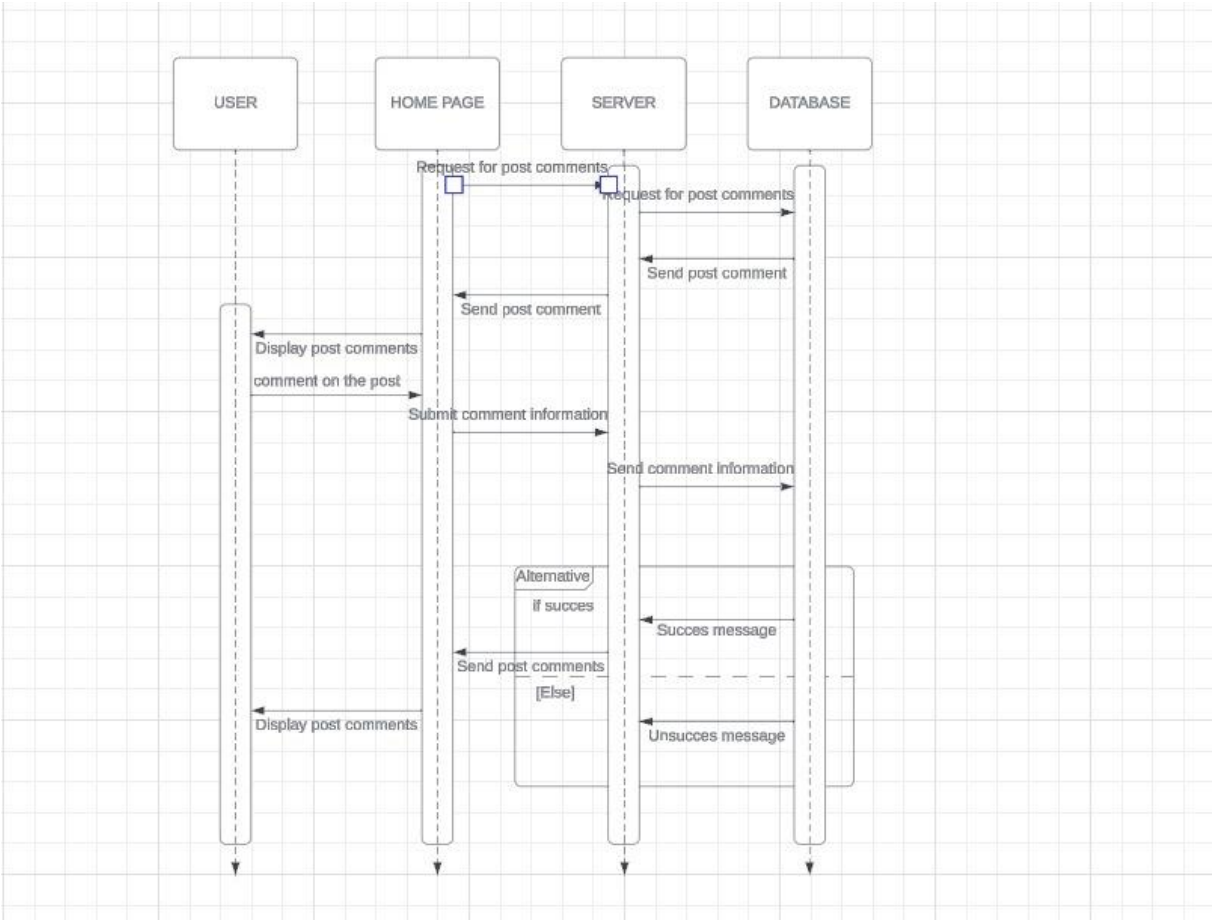

User Logs in
User Navigates to create post sreen
Yes
Add something to the list
List Does Not exists
Create a new list
List exists
Chose Media type
Series
Movie
Game
No
Selected media added to the list

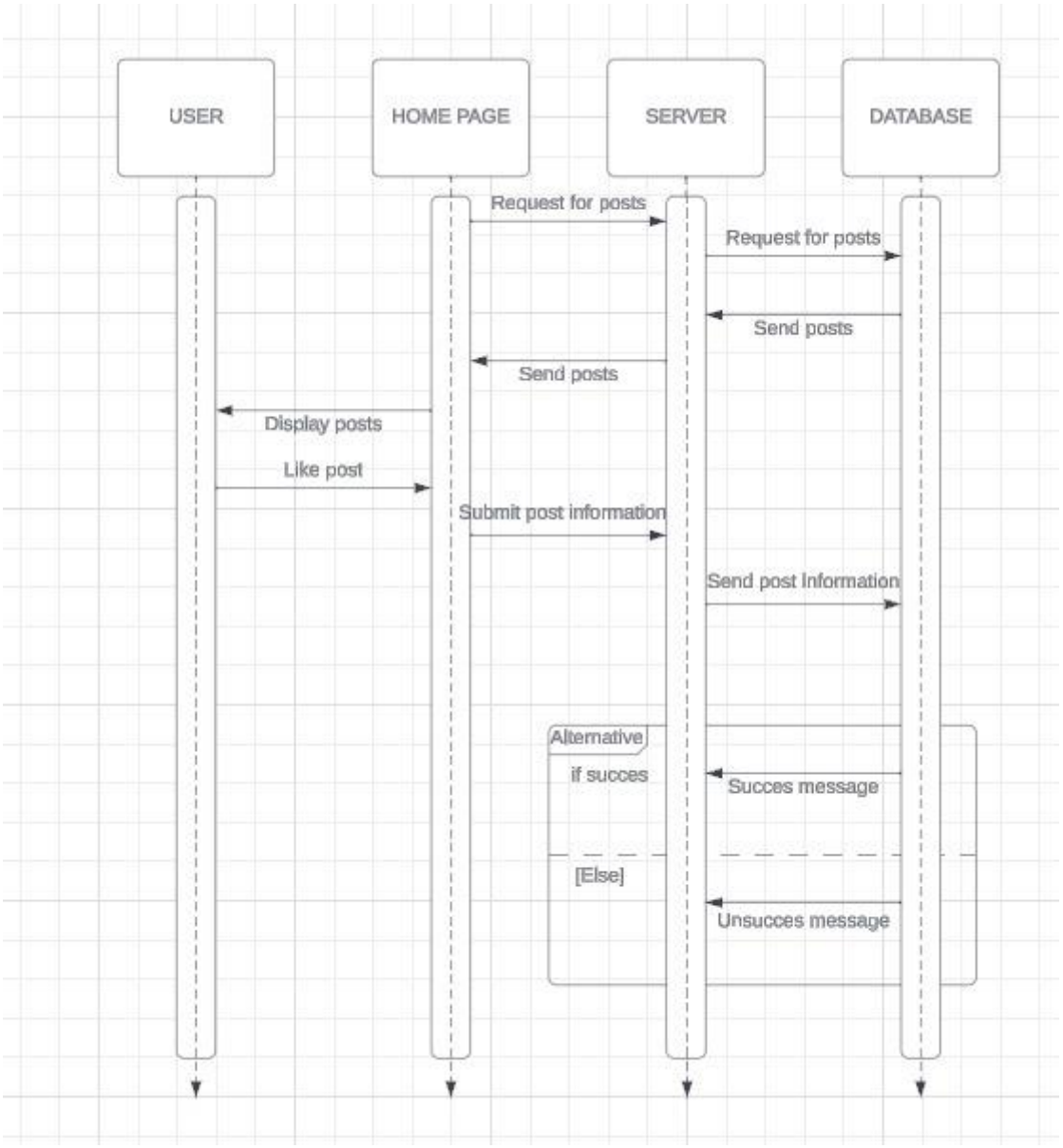

User Logs in
User Navigates in feed
user likes a post
does user wants to see liked posts
Yes
No
User Navigates to profile page

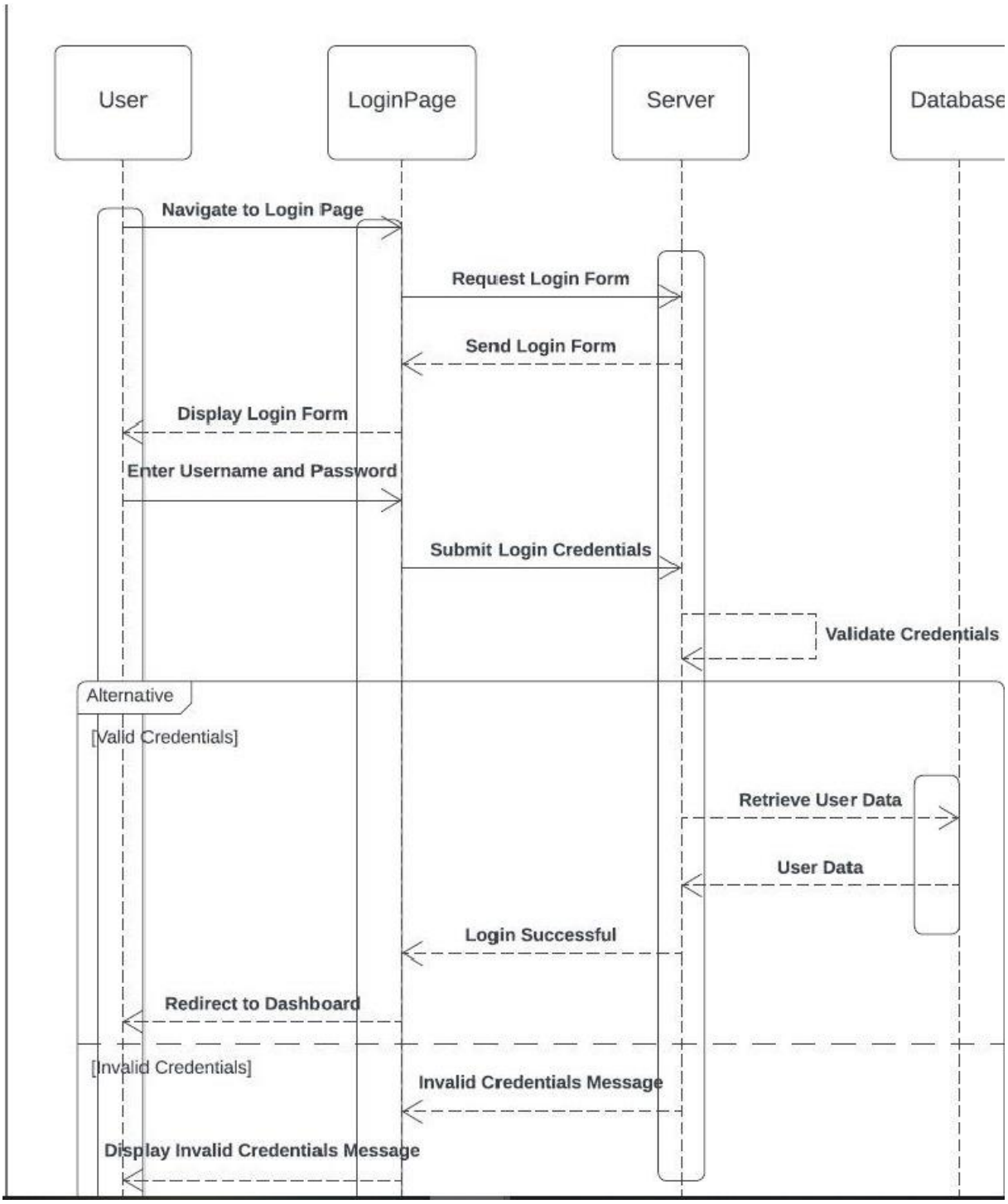

User Navigates to login page
Post something?
No
Yes
View Feed?
Chose post type
Series
Movie
Game
Yes
Post Created
Display Feed

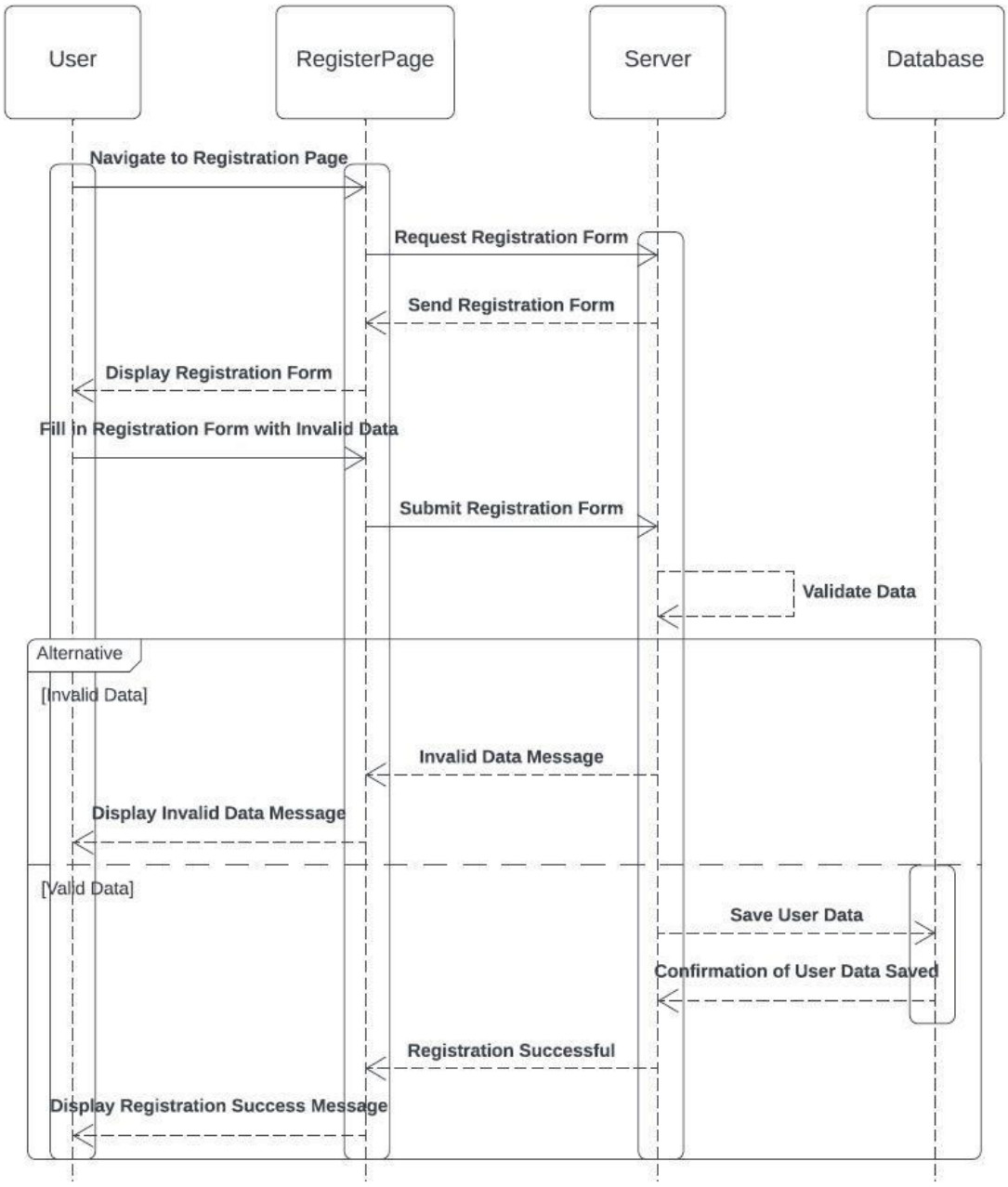

User Logs in
User Navigates to spesific media page
User writes a comment to comment section
is comment appropriate
Yes
No
Server Adds the comment
Error

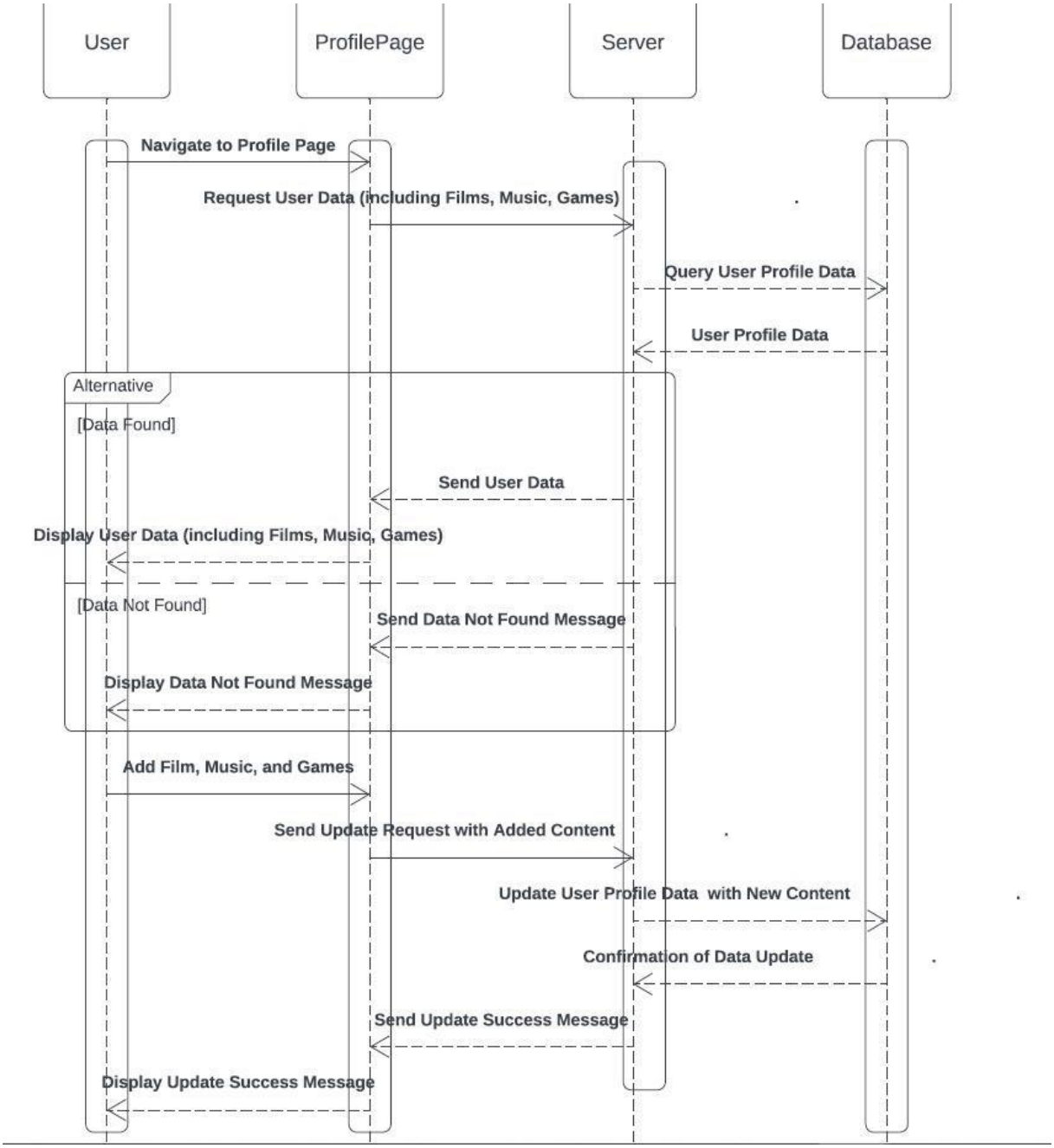
SEQUENCE DIAGRAMS:

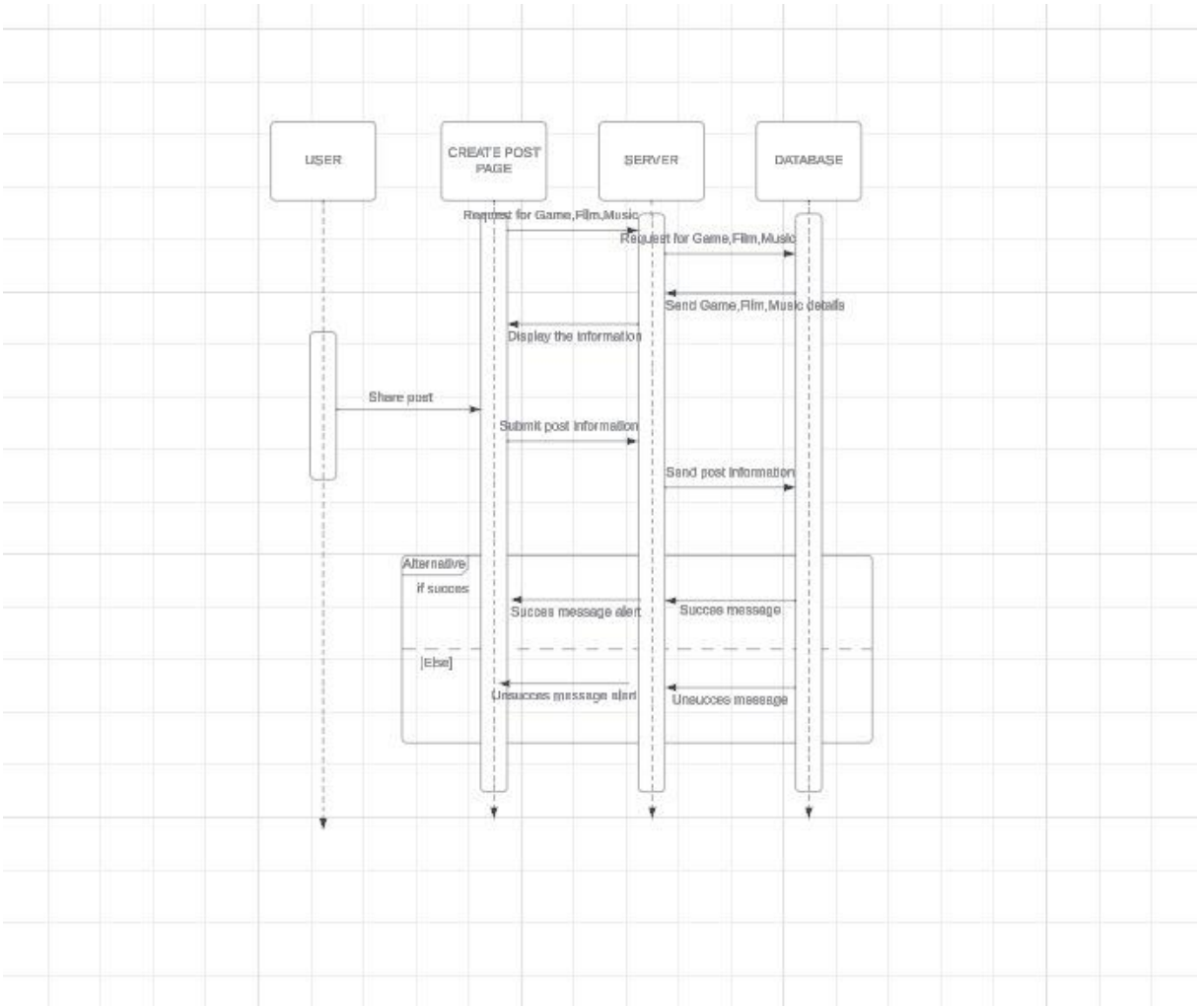



USER
HOME PAGE
SERVER
DATABASE
Request for posts
Request for posts
Send posts
Send posts
Display posts
Like post
Submit post information
Send post Information
Alternative
if succes
Succes message
[Else]
Unsucces message

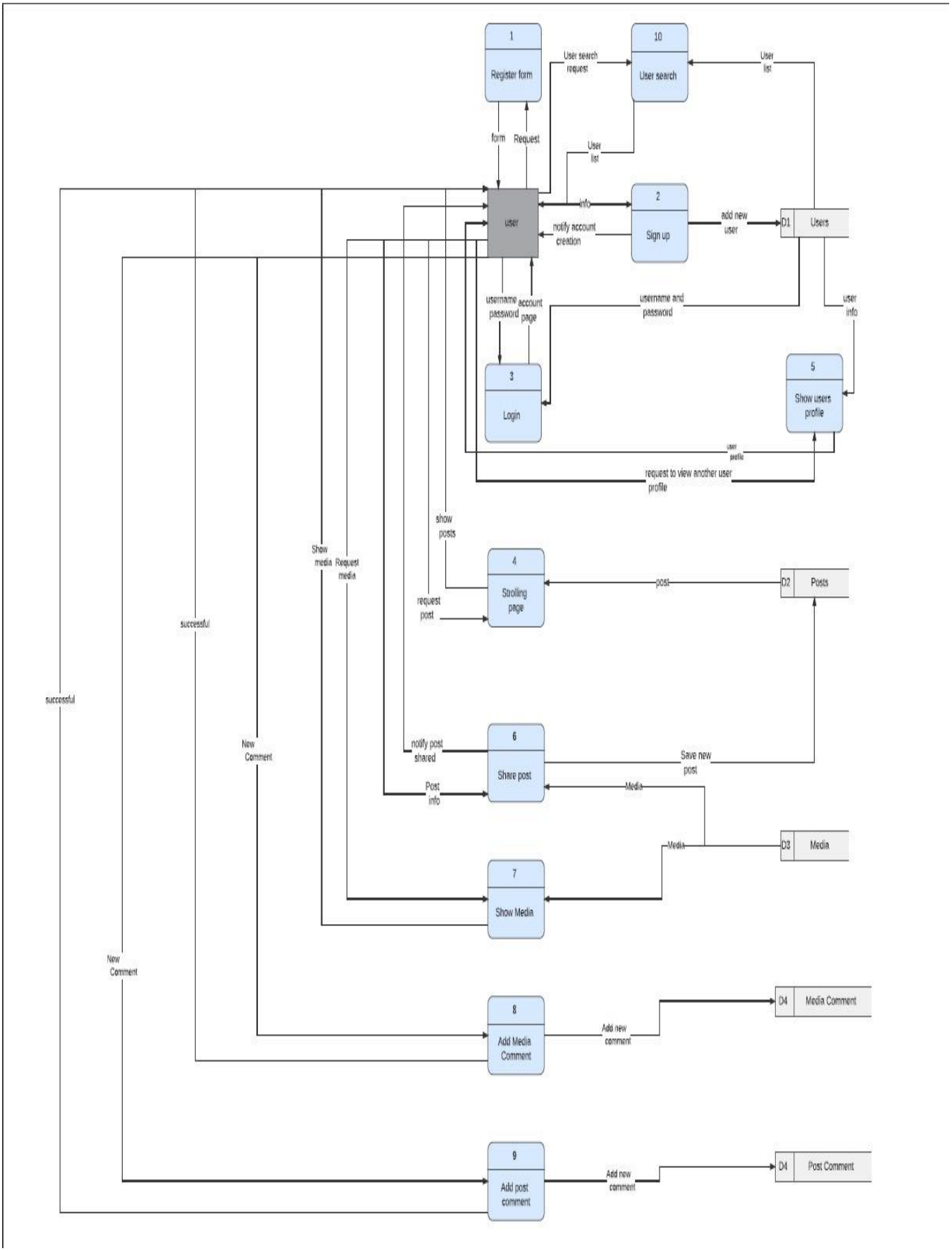

User
LoginPage
Server
Database
Navigate to Login Page
Request Login Form
Send Login Form
Display Login Form
Enter Username and Password
Submit Login Credentials
Validate Credentials
Alternative
[Valid Credentials]
Retrieve User Data
User Data
Login Successful
Redirect to Dashboard
[Invalid Credentials]
Invalid Credentials Message
Display Invalid Credentials Message


User
RegisterPage
Server
Database
Navigate to Registration Page
Request Registration Form
Send Registration Form
Display Registration Form
Fill in Registration Form with Invalid Data
Submit Registration Form
Validate Data
Alternative
[Invalid Data]
Invalid Data Message
Display Invalid Data Message
[Valid Data]
Save User Data
Confirmation of User Data Saved
Registration Successful
Display Registration Success Message


User
ProfilePage
Server
Database
Navigate to Profile Page
Request User Data (including Films, Music, Games)
Query User Profile Data
User Profile Data
Alternative
[Data Found]
Send User Data
Display User Data (including Films, Music, Games)
[Data Not Found]
Send Data Not Found Message
Display Data Not Found Message
Add Film, Music, and Games
Send Update Request with Added Content
Update User Profile Data with New Content
Confirmation of Data Update
Send Update Success Message
Display Update Success Message


USER
CREATE POST PAGE
SERVER
DATABASE
Request for Game,Film,Music
Request for Game,Film,Music
Send Game,Film,Music details
Display the information
Share post
Submit post information
Send post information
Alternative
if succes
Succes message alert
Succes message
[Else]
Unsuccess message alert
Unsuccess message

Data Flow Diagram:



USE CASE DIAGRAM