Department of Computer Engineering

CS353 Term Project

CS353-7-MSDMS

Fall / 2020

Final Report

Team

Talha Şen 21702020
Hakan Sivuk 21601899
Cevat Aykan Sevinç 21703201
Yusuf Nevzat Şengün 21601720

Instructor: Özgür Ulusoy

Teaching Assistants: Arif Usta, Mustafa Can Çavdar

Project Teaching Assistant: Arif Usta

Table of Contents

1. Application Description

This project is an implementation of a media services data management system (MSDMS, such as Netflix) for applying the theory learned in the course and observing real database design practises. Essentially, the project simulates a Netflix like platform where users can interact with media. There are two types of users: a normal user who could interact with the media and an admin user who can create or manipulate media.

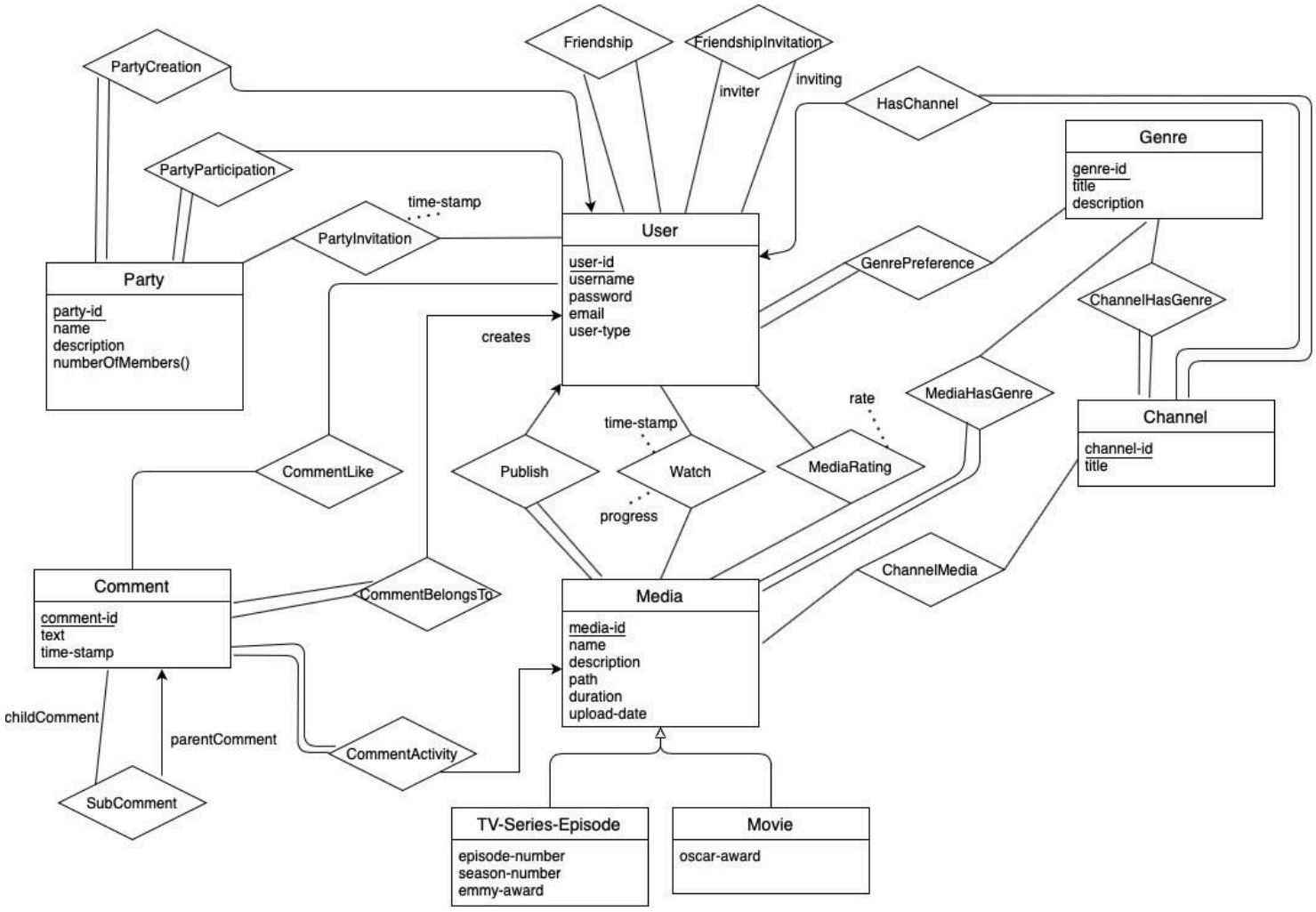
A normal user, after signing up or logging in, can watch media and rate them. They can create channels to save their media. They could search for any media they want to watch and sort them according to their preference of date interval, genre, name or the time they are created. They can add other users as friends. Users can state their genre to get a variety of media suggestions on their channels or while watching media. They can leave feedback on any media they watch. Users can also create parties. They can invite their friends to the party. This way, users can watch the media together with their friends.

An admin user can create and upload a new movie or a new tv show/episode as new media. They can also edit existing media created by them or delete them. Admin users, as if a normal user, can search their media created by them and sort them accordingly.

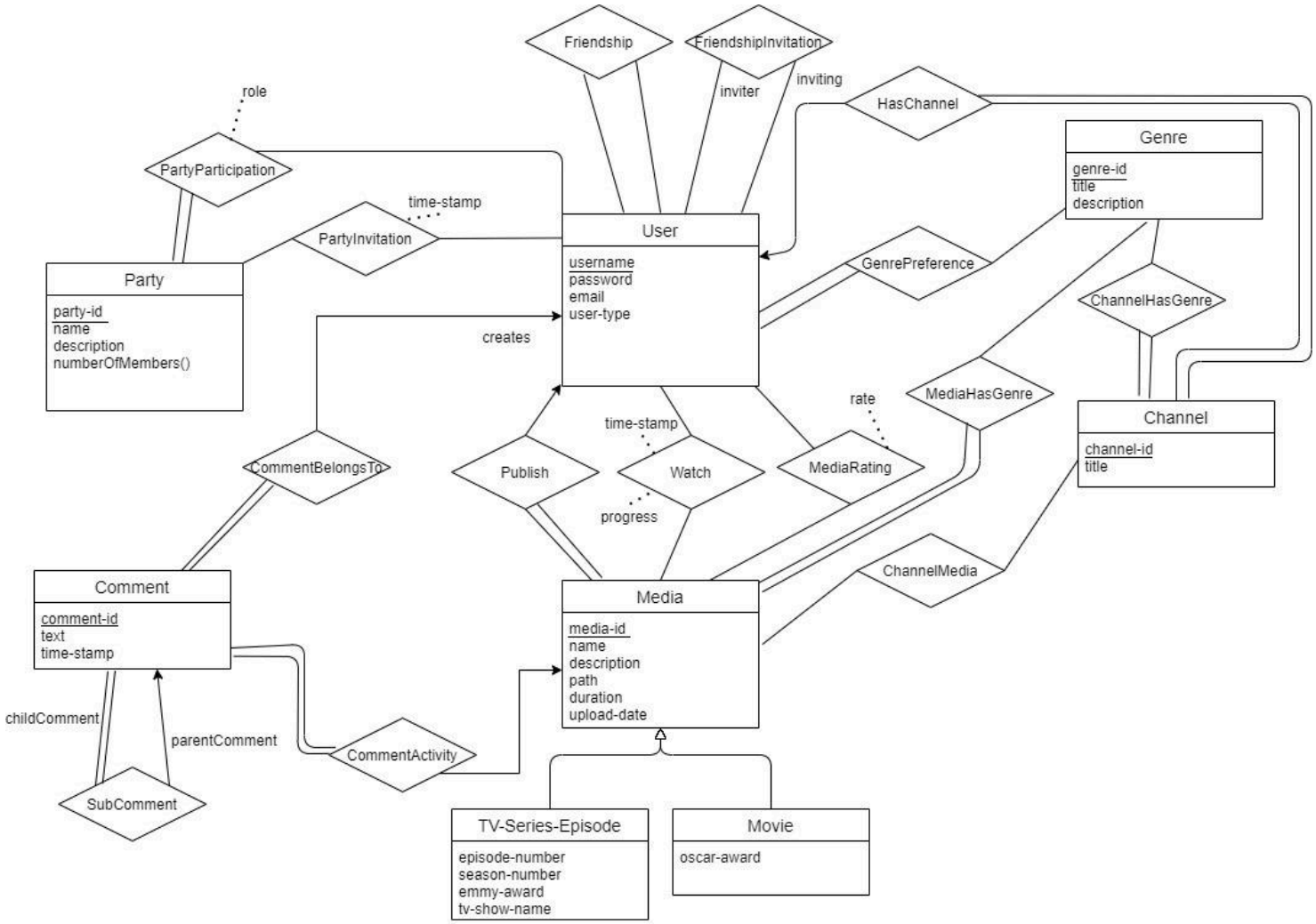
2. Final Database Design

2.1. Final E/R Diagram

Previous (Final E/R diagram is on the next page):



Final:



2.2. Final Table Schemas

Party(<u>party-id</u>, name, description, numberOfMembers)

User(<u>username</u>, password, email, user-type)

Genre(<u>genre-id</u>, title, description)

Comment(<u>comment-id</u>, username, media-id, text, time-stamp)
username is foreign key referencing User(username)
media-id is foreign key referencing Media(media-id)

Media(media-id, publish-user-name, name, description, path, duration, upload-date)
publish-user-name is foreign key referencing User(username)

TV-Series-Episode(media-id, episode-number, season-number, emmy-award, tv-show-name)
media-id is foreign key referencing Media(media-id)

Movie(media-id, oscar-award)
media-id is foreign key referencing Media(media-id)

Channel(channel-id, username, title)
username is foreign key referencing User(username)

SubComment(child-id, parent-id)
parent-id is foreign key referencing Comment(comment-id)
child-id is foreign key referencing Comment(comment-id)

GenrePreference(username, genre-id)
username is foreign key referencing User(username)
genre-id is foreign key referencing Genre(genre-id)

MediaHasGenre(media-id, genre-id)
media-id is foreign key referencing Media(media-id)
genre-id is foreign key referencing Genre(genre-id)

ChannelHasGenre(channel-id, genre-id)
channel-id foreign key referencing Channel(channel-id)
genre-id is foreign key referencing Genre(genre-id)

ChannelMedia(media-id, channel-id)
media-id is foreign key referencing Media(media-id)
channel-id foreign key referencing Channel(channel-id)

Watch(username, media-id, progress, time-stamp)
username is foreign key referencing User(username)
media-id is foreign key referencing Media(media-id)

MediaRating(<u>username, media-id</u>, rate)
username is foreign key referencing User(username)
media-id is foreign key referencing Media(media-id)

PartyParticipation(<u>party-id, username</u>, role)
party-id is foreign key referencing Party(party-id)
username is foreign key referencing User(username)

PartyInvitation(<u>party-id, username</u>, time-stamp)
party-id is foreign key referencing Party(party-id)
username is foreign key referencing User(username)

Friendship(<u>friend1-id, friend2-id</u>)
friend1-id is foreign key referencing User(username)
friend2-id is foreign key referencing User(username)

FriendshipInvitation(<u>inviter-id, invited-id</u>)
inviter-id is foreign key referencing User(username)
invited-id is foreign key referencing User(username)

3. Implementation Details

3.1. Languages, Libraries, Frameworks Used

- **Frontend:** We used the React library to build the frontend. We especially used React Hooks through almost every React component.
- **Backend:** We used Node.js to set up the server. Our server is RESTful. When a request comes to the server from the frontend, the server makes the query to the database.
- **Database:** We are directly using the database that was given us for the programming assignment. The account belongs to Hakan Sivuk. This allowed us to work on the project remotely as we were all able to access the database from our locally running servers. We used the same procedure in the programming assignment to access & manipulate the database.

3.2. Problems Encountered and How They are Solved

We experienced a problem with synchronizing notification fetching**.** Party and friendship notification fetches were cancelling each other. Both could not be viewed at the same time in notifications as their fetch requests raced with each other to update the notifications. We synchronized fetches by creating a new state as fetch received. Once a fetch receives a response, they update the fetch received state and this state updates the previous state of the notifications.

3.3. Functionality Distribution

- **Talha Şen** implemented: Common functionalities and project specific additional requirements, which is the party feature for our project.
- **Hakan Sivuk** implemented: MSDMS functionality 1: Watching a media and its required sub functionalities.
- **Cevat Aykan Sevinç** implemented: MSDMS functionality 3: Publish a media file and its required sub functionalities.
- **Yusuf Nevzat Şengün** implemented: MSDMS functionality 2: Leaving feedback on a media file and its required sub functionalities.

4. Advanced Database Features/Components

4.1. Reports

Total Media Related Report: Gets the general statistics about the media of the system.
SELECT (SELECT TVShowEpisode.tv-show-name FROM Media LEFT JOIN TVShowEpisode ON Media.media-id = TVShowEpisode.media-id UNION SELECT Movie.name FROM Media LEFT JOIN Movie ON Media.media-id = Movie.media-id) as name, AVG(MR.rate) as average-rate, COUNT(DISTINCT W.username) as total-watchers, COUNT(DISTINCT C.comment-id) as total-comments, COUNT(DISTINCT CB.username) as total-commenters
FROM Media M, Watch W, MediaRating MR, CommentActivity C, CommentBelongsTo CB
WHERE M.media-id = W.media-id AND M.media-id = MR.media-id AND M.media-id = C.media-id AND C.comment-id = CB.comment-id
GROUP-BY name
Total User Related Report: Gets the general statistics about the users of the system.

SELECT COUNT(DISTINCT U.username) as total-users, COUNT(DISTINCT W.media-id) as total-watched-medias, AVG(MR.rate) as total-average-rate, COUNT(DISTINCT CB.username) as total-comments, COUNT(DISTINCT PP.username) as total-parties-created, COUNT(DISTINCT F.friend1-id) as total-friends
FROM User U, Watch W, MediaRating MR, CommentBelongsTo CB, PartyParticipation PP, Friendship F
WHERE U.username = W.username AND U.username = MR.username AND U.username = CB.username AND U.username = PP.username AND PP.role = 'creator' AND U.username = F.friend1-id

4.2. Views

We have a Watch table that keeps records of users' watch history. By using this table, the website allows users to be able to continue with their last progress in movies and series. This table is also used for showing friends' activities. This task only requires the last watch activity of each user. Instead of using the *Watch* table, therefore, we are using a view of it for showing friends' activities. *LastActivity* view contains the last watch activity of each user as it is stated above and it is updated once a new activity is added to the Watch table. It allows us to make more efficient and easier SQL operations for the task.

4.3. Triggers, Constraints, Stored Procedures, Secondary Indices

- **Triggers:** We are using a trigger to calculate the number of media in a channel. When a channel is created, its media count is initialized as 0. Then each time a media is added to the channel, the trigger procedure *increment_media_count* is executed and media count is incremented by 1. Also, when we remove a media from a channel, the trigger procedure *decrement_media_count* is executed and media count is decremented by 1. These two procedures update mediaCount column of the Channel, when it is necessary.

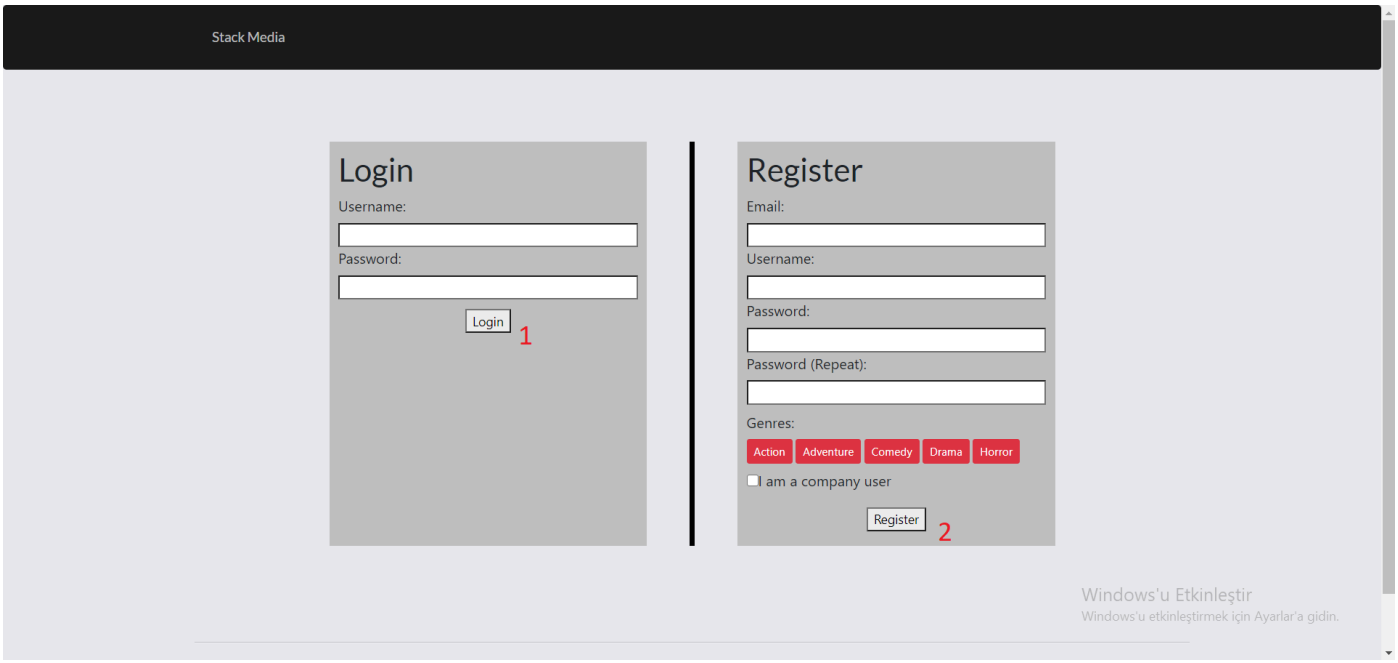
- **Constraints:** We are using foreign keys in tables. Some of these foreign keys require constraints with columns on the referenced table. Therefore, we are using *on delete cascade* and *on update cascade* constraints. Also, we have constraints on the *role* column of the *Party* table and the *userType* column of the *User* table. The *role* column can be either "ROLE_PARTICIPANT" or "ROLE_CREATOR" and the *userType* column can be either "ROLE_ADMIN" or "ROLE_USER". By using constraints, we prevent the possibility of incompatible values.

- **Stored Procedures**: We are using stored procedures to prevent writing an sql code for different tasks repeatedly. Instead, we are storing these sql codes as procedures and execute these procedures when it is necessary. For example deleting a user from FriendshipInvitation table is required for two events. When a user accepts a friendship invitation, we should first remove this entry from FriendshipInvitation table and then add this friendship entry to Friendship table. When a user declines a friendship invitation, we should just remove this entry from the FriendshipInvitation table. Therefore, storing this removing sql code as a procedure allows us to execute it when it is necessary without writing repeated codes. Similarly, we have a stored procedure for removing an entry from the PartyInvitation table.

- **Secondary Indices:** We are using the tv-series name to fetch tv-series and their episodes. Normally, our primary keys are mediaIDs. Our project made heavy use of tv-series name field. For example, we were able to query every tv-show in the system by using tv-serie name. Moreover, we were able to accumulate every season and episode of the series to be watched when a user clicked to watch a tv-show. By making, tv-series name secondary-indice, we made the query process more efficient. This also allowed the system to be more efficient when using aggregation functions to create reports, or get average ratings, etc.

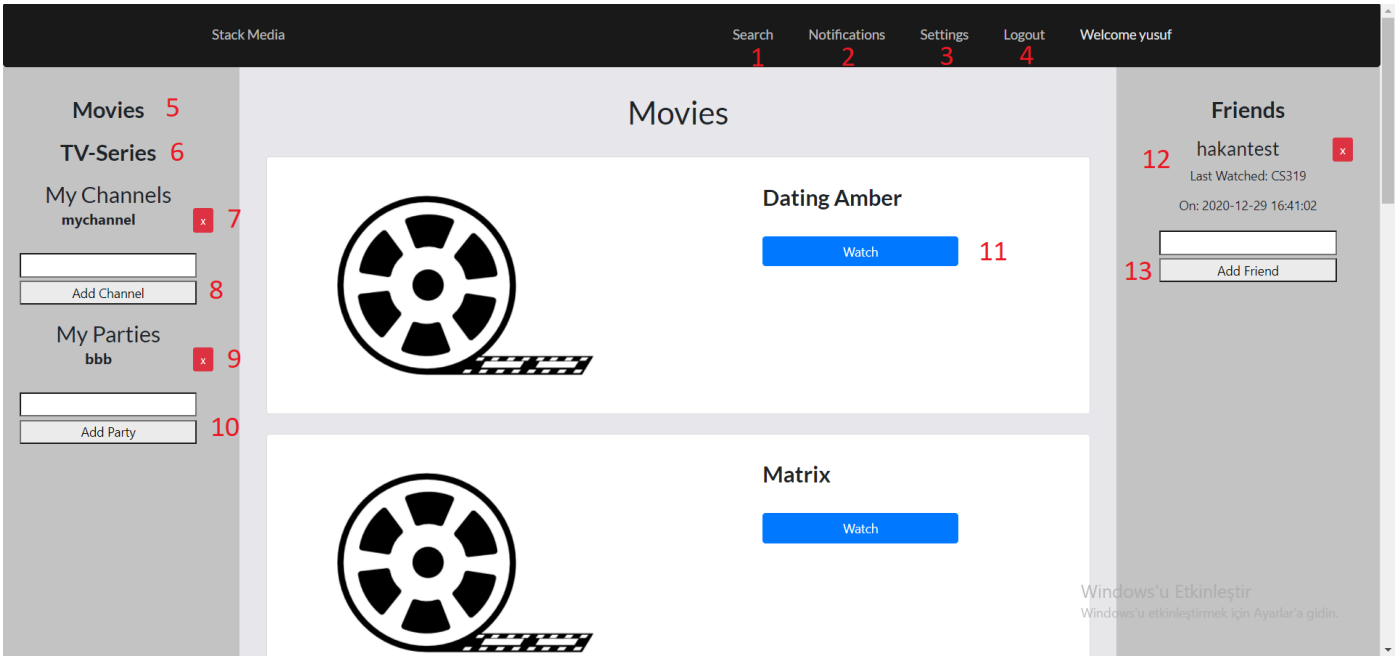
5. User's Manual

5.1. Sign in & Sign up Page (Both Users)



1. By filling username and password, both users can press login to sign in.
2. By filling down the email, username and password fields, both users can register for the application. If a user wants to become a company user and upload media, they have to tick the “I am a company user” box. Company users do not have to pick any genre. However, if a normal user does not pick at least one genre, they are not allowed to register.

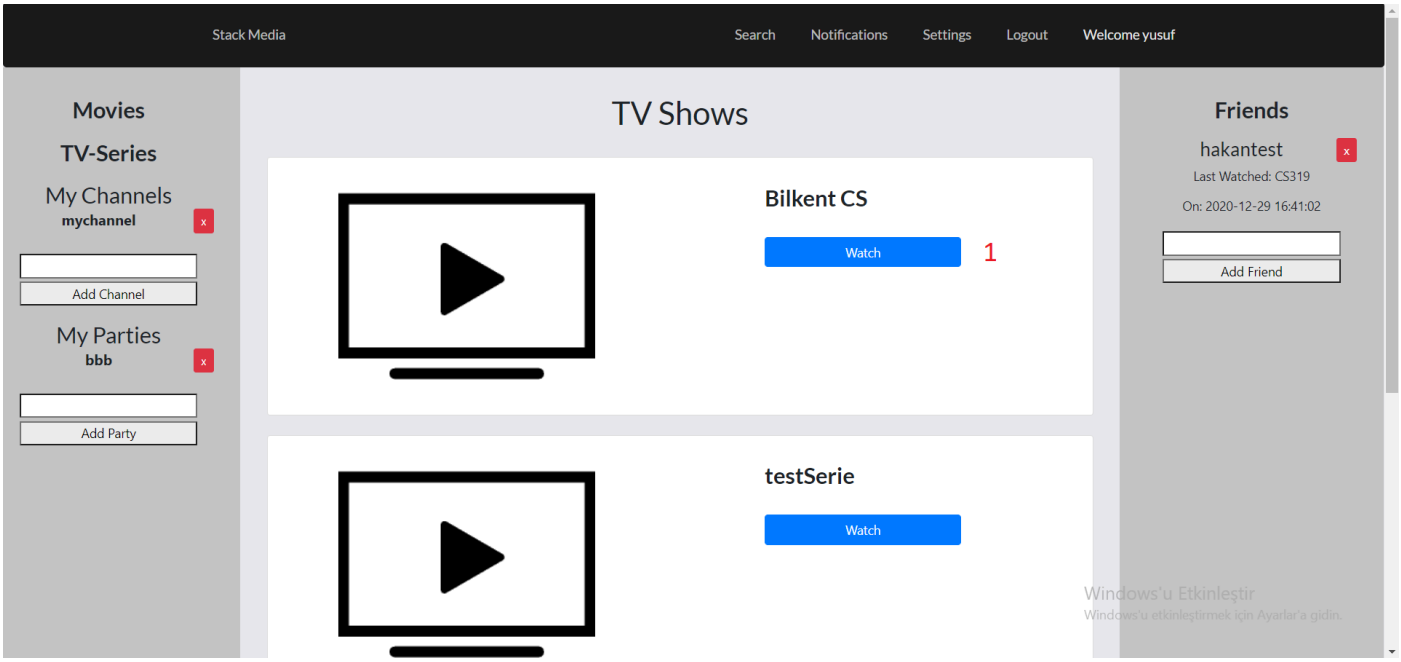
5.2. Main Page Outer Layer and Movies Inner Layer (Normal User)



1. Users can press “Search” to be redirected to the search page where media can be searched in terms of name and sorted according to the genre, name, upload date.
2. Users can press “Notifications” to check if they have received any party or friendship request. Once pressed, a popup window appears that lists any received invitation request. Users could either accept or reject an invitation on this popup window.
3. “Settings” is where password and genre preference could be changed by the user. The user is redirected to the settings page once clicked.
4. By pressing “Logout”, the user logouts and the current session ends. The user would be redirected to the sign in / sign up page.
5. “Movies” is where users can click to view movie media on the inner main page according to their genre preference.
6. “TV-Series” is where users can click to view tv-shows on the inner main page according to their genre preference,
7. Users’ channels are listed here. Users can click on their channel name to open their channel’s page to view any media added by them. Moreover, users can click on the red button to delete any channels of their own.
8. By filling the input field, users can click the “add channel” button to create any channel with the name they state.
9. Users’ parties are listed here. Users can click on the party name to open their parties to watch media together with other party members. By pressing the red button, users can delete any party they are a member of.

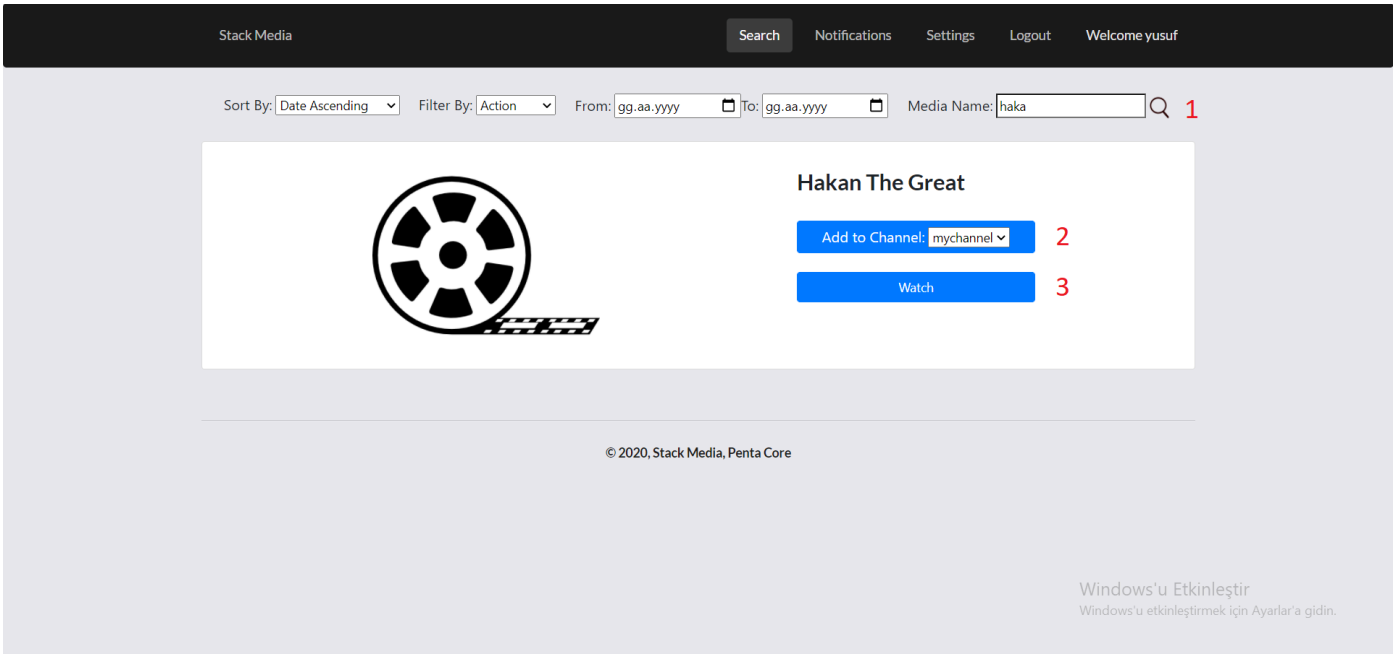
10. By filling the input field, users can click the “add party” button to create any party with the name they state.
11. In the Movies page, movie media according to the genre preference of the user are listed. By clicking on the watch button, the user is redirected to the media page where they can watch the media.
12. Every friend of the user is listed under the Friends. By clicking on the red button, the user can delete any of their friends.
13. By filling the input field, users can click the “add friend” button to send a friendship request to the user with the name stated.

5.3. TV-Series Inner Layer (Normal User)



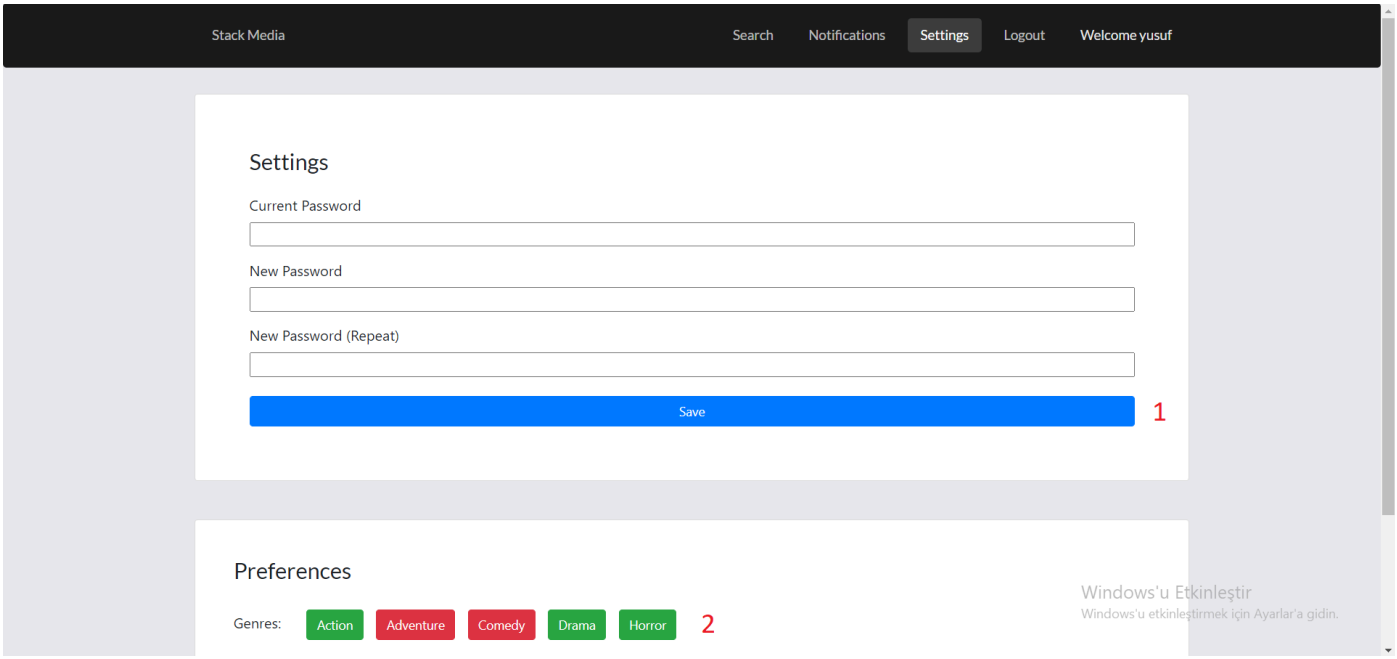
1. In the Tv-Series page, tv series are listed according to the genre preference of the user.. By clicking on the watch button, the user is redirected to the media page where they can watch the media. By default, the last episode of the tv-series will be loaded. If they have not watched the tv-series yet, this will be the first episode. If they have watched and finished a tv-series episode, this would be the next episode of the series.

5.4. Search Page (Normal User)



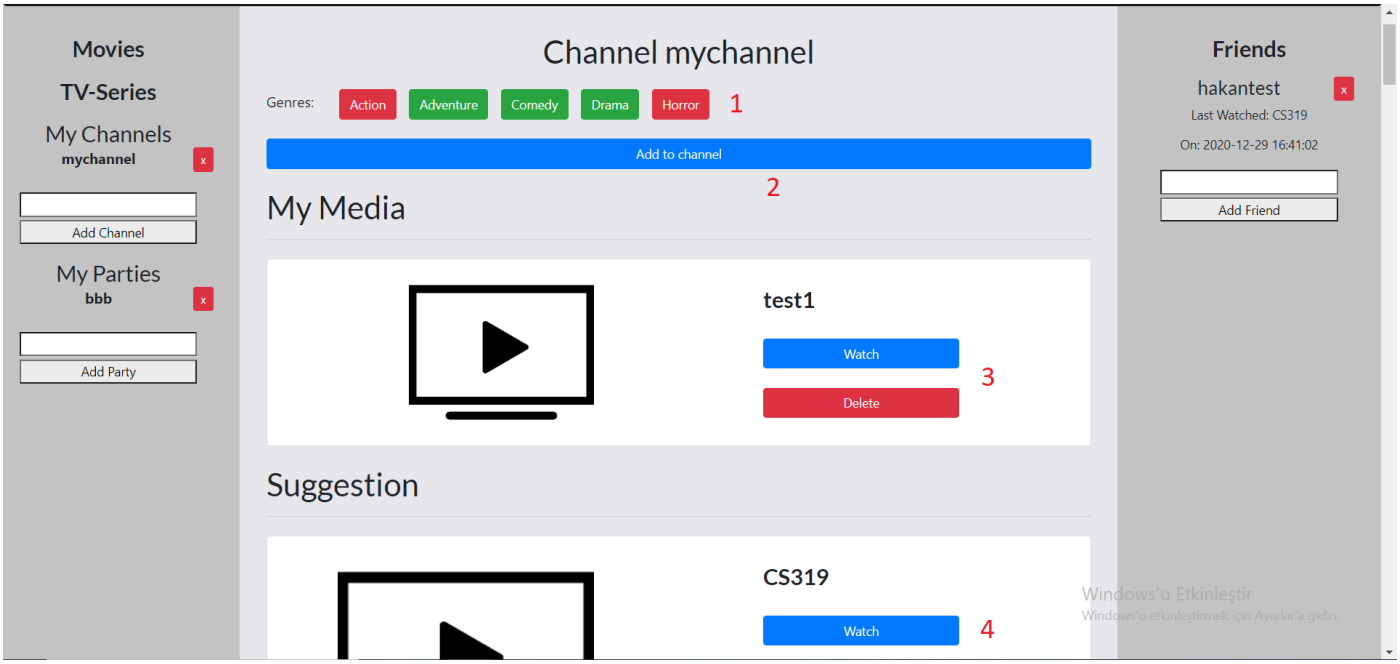
1. This is the bar where users can select their search criterion and search for any media. Users can fill the media name field to search for their media. Any media that is matched with the stated input as a prefix, is displayed to the user. Users can sort the found media by their upload date ascending/descending or by their name ascending/descending. Moreover, users may choose a genre to filter media. Default genre is action. Lastly, users can determine a range to filter media that is uploaded between the given range.
2. Users can choose any channel of their own to add found media to their channels of choice.
3. Users can click the watch button to be redirected to the media page where they can watch the selected media.

5.5. Settings Page (Normal User)



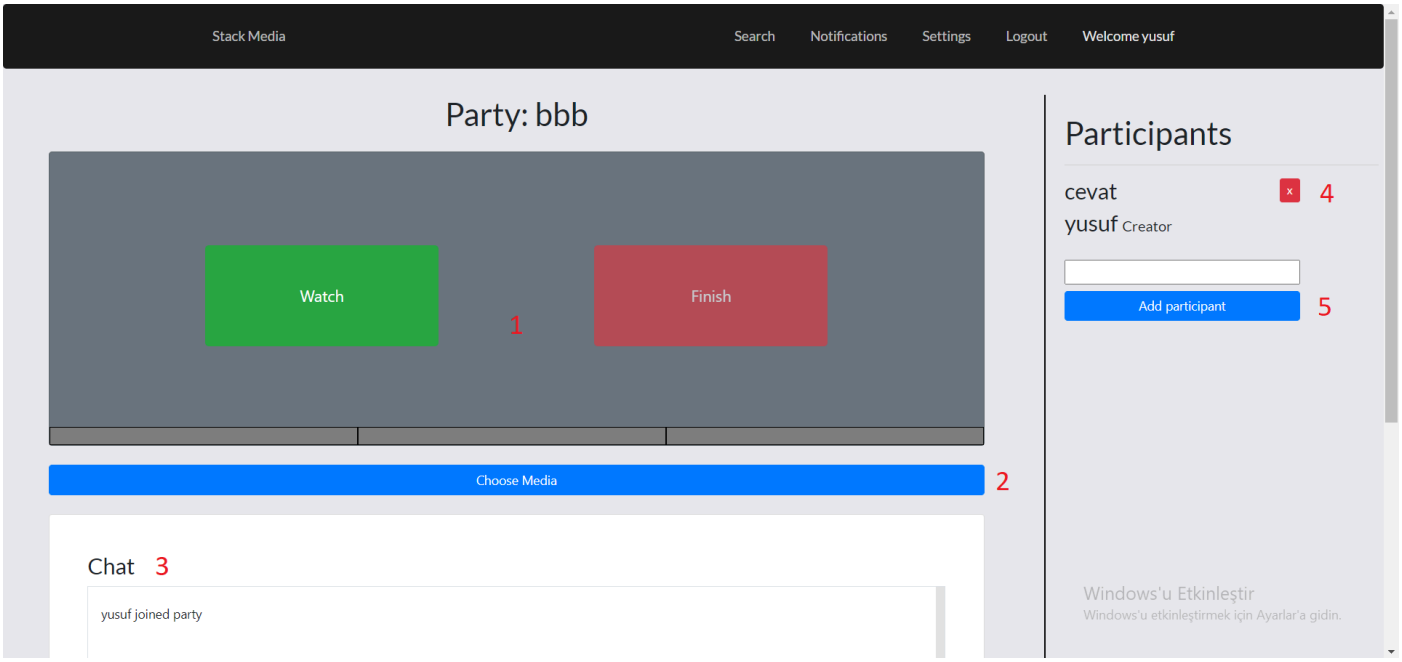
1. Users can change their passwords by indicating their current password, then the new password and its repetition. By clicking the “save” button, the users can request for their passwords to be changed, if the operation is successful, the user is notified with an alert.
2. Users could edit their genre preference here. Genre marked with green states that it is currently chosen. Genre marked with red indicates that the genre is not selected. Users have to have at least one genre. They cannot unselect a genre if it is the only one left for their genre preference. Users can click on the green genre to deselect it and red genre to select it as their preferred genre.

5.6. Channels Page (Normal User)



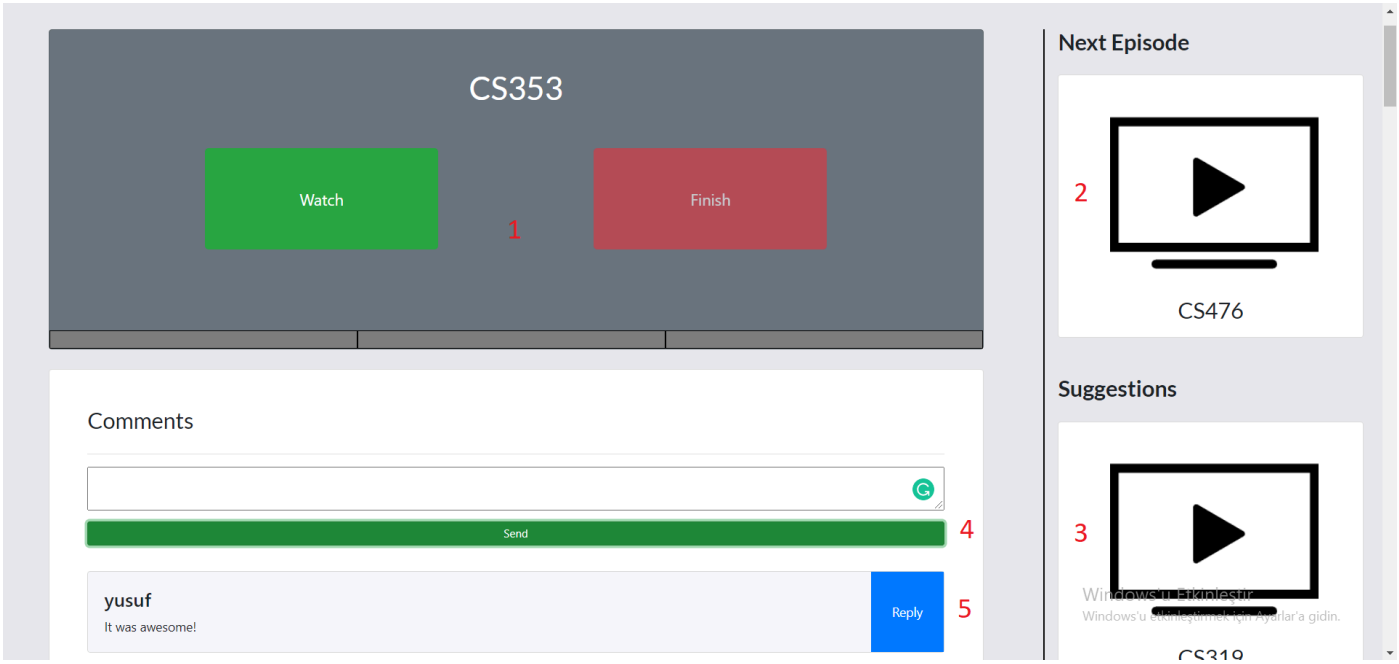
1. Users can select the preferred genre suggestions for the channel. Genres marked with green would be suggested to the user under the suggestions section. Users can click the red genre to select them and green genre to deselect them. Each channel starts with an empty genre. However, if any genre is selected, at least one genre preference must be maintained for the channel and this is constrained.
2. Users can press the “add to channel” button to be redirected to the search page where they can search for media to add to their preferred channels.
3. Users can view their channel media under the my media section. Users can press the “delete” button to delete any media they had added to their channel previously. Next, they can press the “watch” button to be redirected to the media page where they can watch the selected media.
4. Any suggested media for the channel based on the channel genre preference are displayed under suggestions. Users can click watch to be redirected to the media page to watch the suggested media.

5.7. Party Page (Normal User)



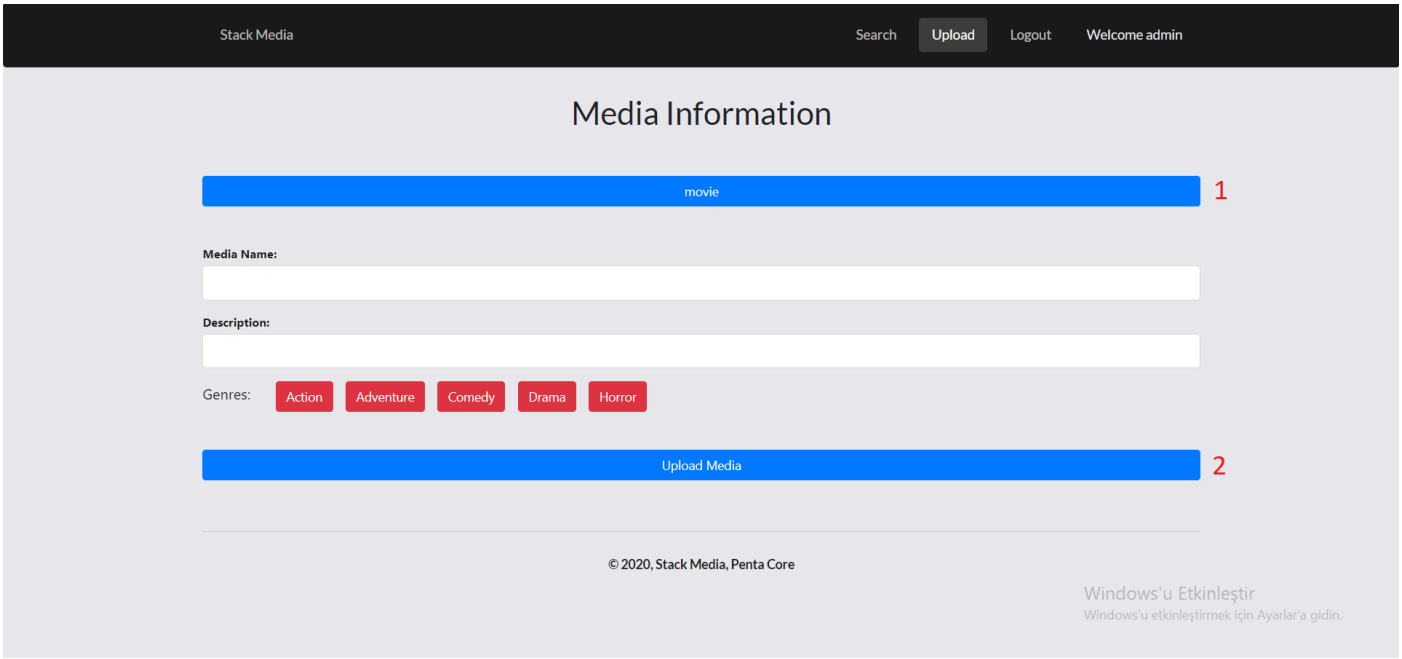
1. Users can press the watch button 3 times and after this the finish button once to watch and finish watching the media along with other party participants that are currently in the party.
2. Only the party creator can choose media to watch with the party participants. By clicking “choose media”, a small search screen appears and just as the search media page, media is searched.
3. Party participants that are currently in the party can communicate by sending messages on the chat.
4. The party creator can remove any party participant from the party by clicking on the “cross button”.
5. By filling the input, party creator can send party invitation to the user they indicate.

5.8. Media Page (Normal User)



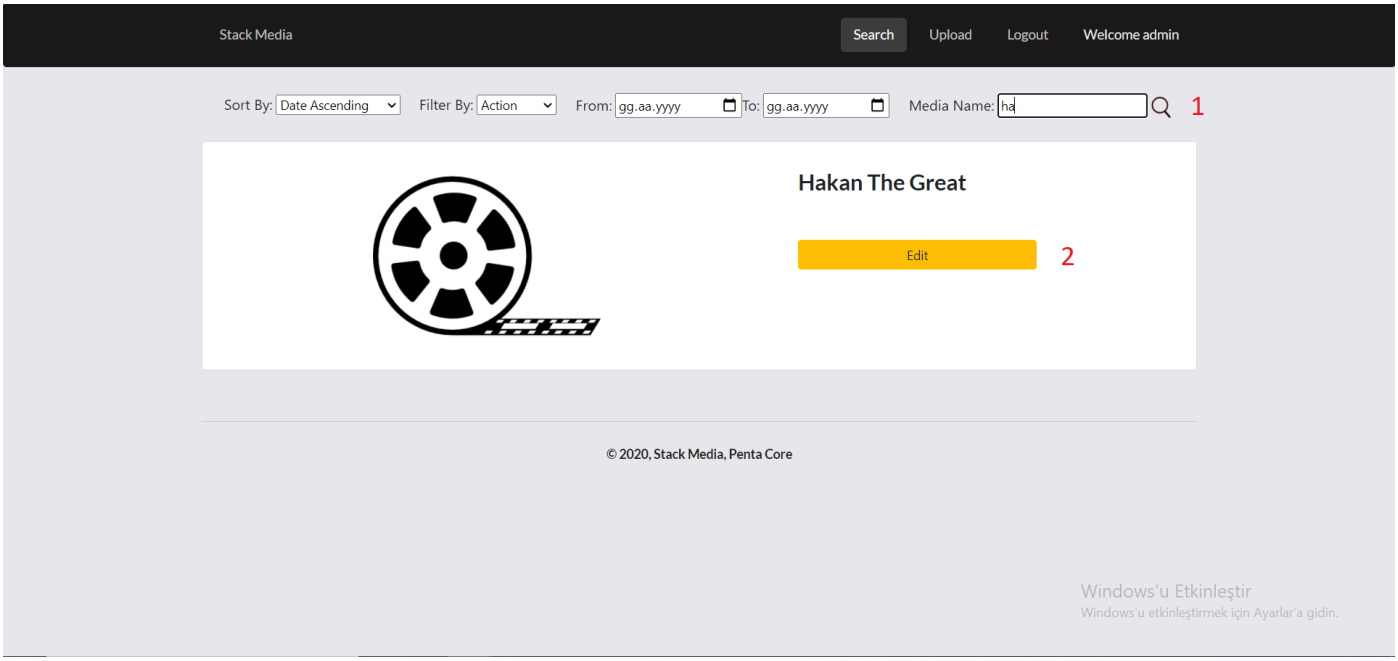
1. Users can press the watch button 3 times and after this the finish button once to watch and finish watching the media. The progress of the user is indicated with the bars under the buttons. The finish button is only available once all 3 bars are full.
2. If the current watched media belongs to a tv series, the next episode to watch appears under “next episode”. If the currently watched media is a movie, this section is not displayed. By clicking on the media icon, users can watch the next episode of the tv series.
3. Suggested media for the watched media appears under “suggestions”. This suggestion is based on the watched media’s genre. By clicking on the media icons, the user can watch any of the suggested media.
4. Users can make comments on the media they watch. They can fill the input field and press the “send” button to send their comments. Comments are sorted by the most recent to the oldest.
5. Users can reply to any comment that is made on the media. By pressing the “reply” button, an input field appears under the comment to make a comment. By filling this input field and pressing the “send” button just under this field, users can make sub-comments. Sub-comments are sorted by the oldest to the most recent.

5.9. Main Page / Upload Media Page (Admin User)



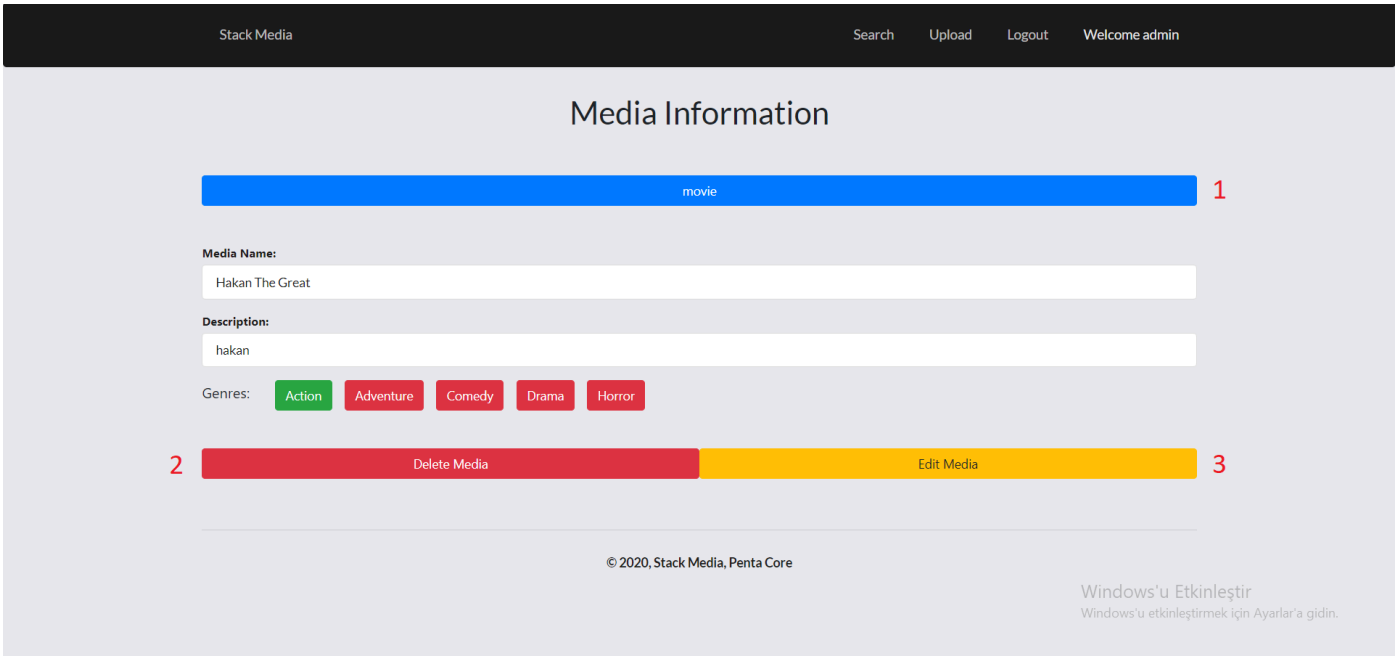
1. Admin users can select the media type they are uploading as either movie or tv show. If it's a movie, they can enter its name and description. If it's a tv show, they can also add its episode name, episode and season numbers. In addition, they must also assign the media they are uploading to at least 1 genre given in the list. If a genre is selected, it will be shown as green, if it is not selected, it will be shown as red.
2. By pressing the “Upload Media” button, admin users upload the media to the system if they entered all the required fields.

5.10. Search Page (Admin User)



1. Admin users can search their previously uploaded media by entering its name to the search box. The search box does a likewise search so from the start each matched character will show any media with matched characters as well, and it builds up after each entered character. They can also filter the search by date/name ascend/descend, genre and date range (“from” date is inclusive and “to” date is exclusive).
2. By clicking the “Edit” button they are redirected to Edit / Delete Media Page to edit the media.

5.11. Edit / Delete Media Page (Admin User)



1. Admin users can select the uploaded media type if it should be a movie or a tv show. If it's a movie, they can change their movie name and description. If it's a tv show, they can also change their episode name, episode and season numbers. Admin users have to specify at least one genre for the media to be successfully uploaded and this is constrained. Green genre indicates the genre is selected and red states that the genre is not selected.
2. Admin users can delete the media from the system completely.
3. After admin users edit the media with the values they want, they can apply their edit options to the related media by pressing this “edit media” button.