Department of Computer Engineering

CS353 Term Project - Stack Media

CS353-7-MSDMS

Fall / 2020

Project Proposal Report

Team

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

Instructor: Özgür Ulusoy

Teaching Assistants: Mustafa C. Çavdar, Arif Usta

Project Teaching Assistant: Arif Usta

Table of Contents

1. Introduction

Stack Media is a media service system developed by group 7. Similar to Netflix, it will allow registered users to watch many movies or TV shows at any given time. Our application will feature several genres with many movies and movie series included in them. Likewise, we will feature different seasons of a TV show if it's a multi-season show. In addition to this, our application will have a social aspect to it. Users may add friends to their friends list and inspect their activity on Stack Media. Users may give feedback on a movie or TV show by giving a comment or review, or liking them. These comments, reviews or likes will be visible to other users. Users may also create channels to follow different movies, similar to a watch list, to watch them later on. Users may also specify their preferences on genres to allow our system to provide suggestions to the user based on their preferences. There will be a search function where users can search movies or TV shows by their names, genres or even actors. Lastly, users will be allowed to host parties. Parties are online movie or TV show live streams where users may add their friends to the party to watch a movie or TV show together from the same video player.

2. Proposed Application

In this section of the report, both functional and non-functional requirements of the project will be discussed. After that, the constraints of the project will be explained.

2.1. Requirements

2.1.1 Functional Requirements

- **Create Account**
  Users must be able to create an account through a registration form to use the functionalities of the website.

- **Login**
  Users must be able to login to their accounts through a login panel in order to authenticate.

- **Logout**
  Users must be able to logout from their accounts through a button.

- **Change User Info**
  Users must be able to change their account's information such as name and email any time after registration.

- **Add/Remove Friend**
  Users must be able to add any user as a friend and remove a user from the friend list.

- **Create Channel**
  Users must be able to create media channels to add desired movies and series. The channel name can be customized when it is created.

- **Add Media to/Remove Media from Channels**
  Users must be to able add movies and series to their channels. Also, they must be able to remove some of them whenever they want.

- **Browse Media**
  Users must be able to browse movies or series through the search bar and navigate to the media page.

- **Like a Media**
  Users must be able to like movies and series.

- **Make Comment on a Media**
  Users must be able to make comments on movies and series.

- **Reply to Comments**
  Users must be able to make comments on other users' comments.

- **Like Comments**
  Users must be able to like other users' comments.

- **Add/Remove Genre as a Preference**
  Users must be able to add or remove genres as a preference. This will affect movies or series that are recommended by the website.

- **See Friends' Activity**
  Users must be able to see their friends' activities on one side of the page. Activities include last watched movies or series.

- **Upgrade Premium Account**
  Users must be able to upgrade their account in order to see the premium contents.

- **Downgrade Ordinary Account**
  Users must be able to downgrade their account from premium when they want to.

2.1.2 Additional Functional Requirements

- **Write Review**
  Users must be able to write review essays. In these texts, for instance, users can make an evaluation of a movie or they can compare two series. A review is written by a user and as it may not be related to any media(movies or series), it may also be related to one or more media.

- **Like Review**
  Users must be able to like other users' review essays.

- **Browse Actor/Actress**
  Users must be able to browse an actor/actress through movies or series he/she plays or through the search bar. Users can see other movies or series he/she plays at the actor's/actress' page.

- **Create Party**
  Users must be able to create parties. In a party, multiple users can watch movies or series simultaneously as they are watching it altogether. Other users will be invited by the owner of the party and also one or more movies or series will be assigned to be watched during the party.

- **Invite User to Party/Accept or Decline Party Invitation**
  Users must be able to invite other users to his or her party. Also, invited users must be able to accept or decline this invitation.

- **Add Media to Party**
  The owner of the party must be able to assign movies or series to the party so that these will be watched during the party.

2.1.3 Non-Functional Requirements

- **Performance**
  Users must be able to browse movies/series and navigate through the pages without any latency that affects user experience in a bad way. Also, while watching the media, there should not be any freezing in the video.

- **Extensibility**
  The project structure must be suitable for any future extension. Since the website is directly related to user experience, extensions may be done according to people's changing demands. Therefore, it is important both database and project design should be extensible.

- **Robustness**
  The frontend, server and database parts of the application should work accordingly to prevent any kind of error. Therefore, the network system should be robust to handle any kind of connection problem.

- **Security**
  User data kept in the database such as passwords must be handled properly.

2.2 Constraints

- **Data**
  To serve customers information, we need to insert information related to media to our database. This can be done manually or related information could be scrapped from the web. The more data in the database, the richer the project.

- **Time**
  The project must be implemented in two months. This means that the number of functionalities could be impacted by time if the business plan is not properly managed.

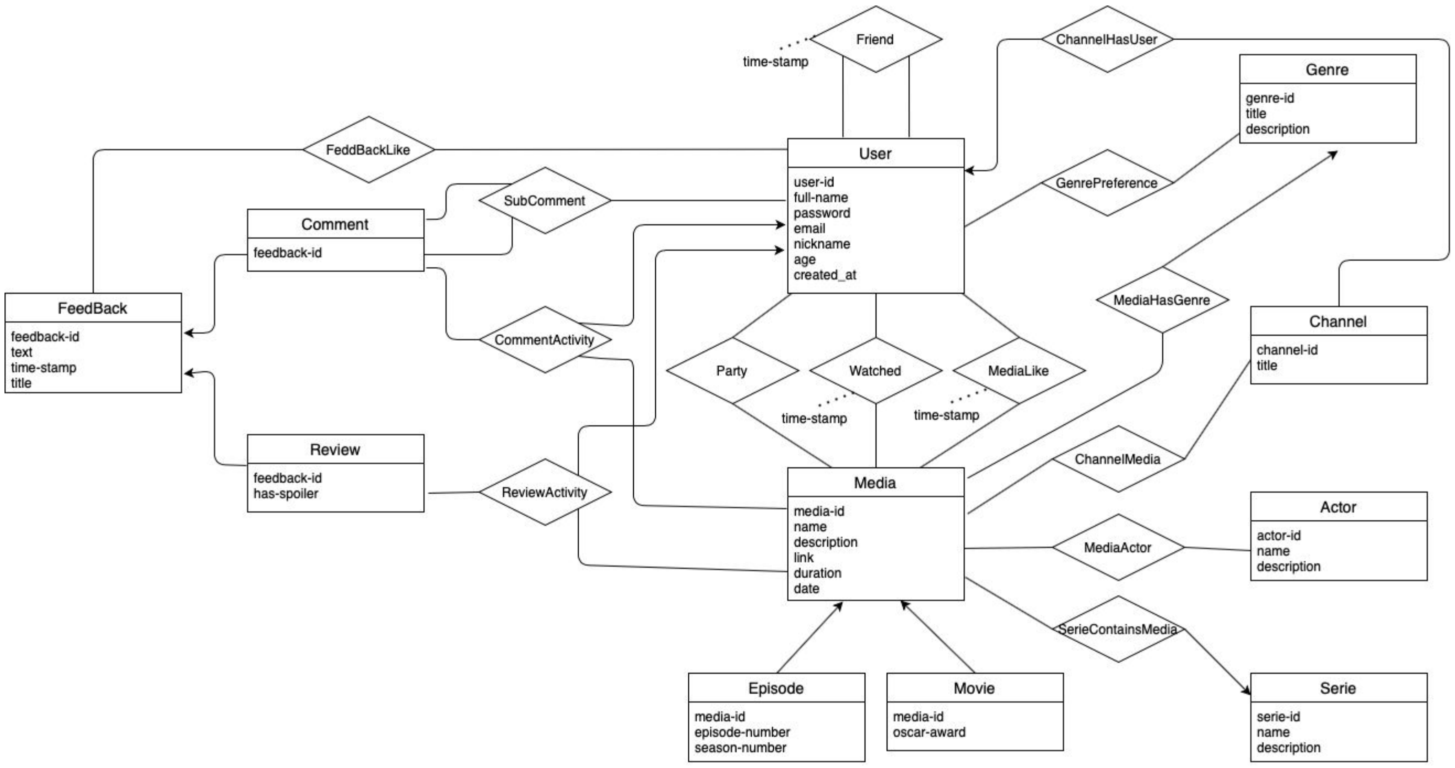
- **Budget**
  As students, we do not have a budget. This indicates that when our server is hosted, our project scalability could depend on the hosting plan according to our budget.

3. Database

A database management system will be needed for our project. The persistent data cannot be kept client-side. This would mean that every information across different users would have to be transformed to each other client at every user activity. Next, if the persistent data is kept in a database for each entity that would persist in the system, a server could serve a client for their requests. This way, persistent data would be processed by the server before being served or updated. Therefore, a database used with a server would make our system more reliable, scalable and secure.

In our project, there are many entities and many relationships between these entities. There could be many relations such as genre preference, a friendship between users, liking a media or writing a review. This information should be represented by distinct relations and entities. As a result, a database will be used to manage object/entity relationships and functionalities related to these relationships in our application. As a result, we plan to use the MySQL database with DBeaver to manage it while React will be used for the frontend.

4. Conceptual Design of the Database



(You may click on the image to access the image file.)

5. Project Website

https://hq-project.github.io/CS353-Website-Project/.