Department of Computer Engineering

CS353 Term Project

CS353-7-MSDMS

Fall / 2020

Design 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. Database Design **2**

1.1. Revised E/R Diagram 2

1.2. Table Schemas 3

1.3. Table Schemas SQL 5

2. UI Design and Corresponding SQL Statements **10**

2.1 Login Page UI 10

2.2 Login Page UI Corresponding SQL Statements 10

2.3 Movies Page UI 11

2.4 Movies Page UI Corresponding SQL Statements 12

2.5 Series Page UI 14

2.6 Series Page UI Corresponding SQL Statements 15

2.7 Channel Page UI 17

2.8 Channel Page UI Corresponding SQL Statements 18

2.9 Search Page UI 21

2.10 Search Page UI Corresponding SQL Statements 21

2.11 Watch Page UI 23

2.12 Watch Page UI Corresponding SQL Statements 24

2.13 Party Page UI 27

2.14 Party Page UI Corresponding SQL Statements 27

2.15 Settings Page UI 30

2.16 Settings Page UI Corresponding SQL Statements 30

2.17 Search Page Company UI 32

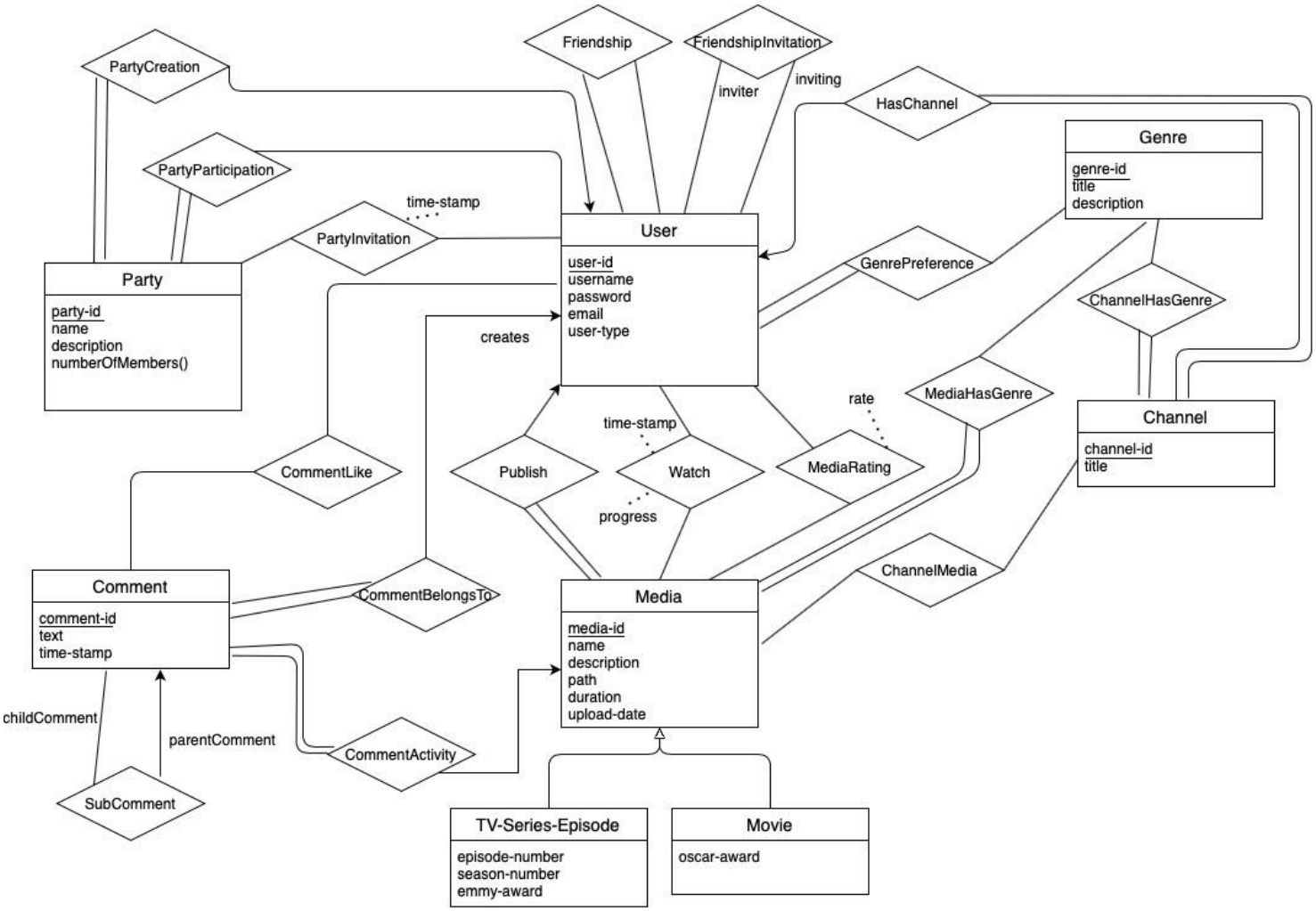
2.18 Search Page Company UI Corresponding SQL Statements 32

2.19 Upload/Edit Media Company Page UI 34

2.20 Upload/Edit Media Company Page UI Corresponding SQL Statements 34

1. Database Design

1.1. Revised E/R Diagram



1.2. Table Schemas

Party(party-id, creator-id, name, description, numberOfMembers)
creator-id is foreign key referencing User(user-id)

User(user-id, password, email, username, user-type)

Genre(genre-id, title, description)

Comment(comment-id, user-id, media-id text, time-stamp)
user-id is foreign key referencing User(user-id)
media-id is foreign key referencing Media(media-id)

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

TV-Series-Episode(media-id, episode-number, season-number, emmy-award)
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, user-id, title)
user-id is foreign key referencing User(user-id)

CommentLike(user-id, comment-id)
user-id is foreign key referencing User(user-id)
comment-id is foreign key referencing Comment(comment-id)

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

GenrePreference(user-id, genre-id)
user-id is foreign key referencing User(user-id)
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(user-id, media-id, progress, time-stamp)
user-id is foreign key referencing User(user-id)
media-id is foreign key referencing Media(media-id)

MediaRating(user-id, media-id, rate)
user-id is foreign key referencing User(user-id)
media-id is foreign key referencing Media(media-id)

PartyParticipation(party-id, user-id)
party-id is foreign key referencing Party(party-id)
user-id is foreign key referencing User(user-id)

PartyInvitation(party-id, user-id, time-stamp)
party-id is foreign key referencing Party(party-id)
user-id is foreign key referencing User(user-id)

Friendship(friend1-id, friend2-id)
friend1-id is foreign key referencing User(user-id)
friend2-id is foreign key referencing User(user-id)

FriendshipInvitation(inviter-id, invited-id)
inviter-id is foreign key referencing User(user-id)
invited-id is foreign key referencing User(user-id)

1.3. Table Schemas SQL

```
CREATE TABLE  Party (
        party-id VARCHAR(32) NOT NULL,
        creator-id VARCHAR (32) NOT NULL,
        name VARCHAR(64) NOT NULL,
        description VARCHAR(256),
        numberOfMembers INT(10),
        PRIMARY KEY(party-id)
        FOREIGN KEY(creator-id) REFERENCES User( user-id) on delete cascade )
ENGINE=INNODB;

CREATE TABLE  User (
        user-id VARCHAR(32) NOT NULL,
        full-name VARCHAR (64),
        password VARCHAR (64) NOT NULL,
        email VARCHAR(128) NOT NULL,
        username VARCHAR(50) NOT NULL,
        user-type VARCHAR(20) NOT NULL,
        PRIMARY KEY(user-id) )
ENGINE=INNODB;

CREATE TABLE  Genre (
        genre-id VARCHAR(32) NOT NULL,
        title VARCHAR(64) NOT NULL,
        description VARCHAR(128),
        PRIMARY KEY(genre-id) )
ENGINE=INNODB;

CREATE TABLE  Comment (
        comment-id VARCHAR(32) NOT NULL,
        user-id VARCHAR(32) NOT NULL,
        Media-id VARCHAR(32) NOT NULL,
        time-stamp TIMESTAMP,
        PRIMARY KEY(comment-id)
        FOREIGN KEY(user-id) REFERENCES User(user-id)
        FOREIGN KEY(media-id) REFERENCES Media(media-id) )
ENGINE=INNODB;

CREATE TABLE  Media (
        media-id VARCHAR(32) NOT NULL,
```

```
        publish-user-id VARCHAR(32) NOT NULL,
        name VARCHAR(64) NOT NULL,
        description VARCHAR(256),
        path VARCHAR(256) NOT NULL,
        duration INT(10) NOT NULL,
        upload-date DATE NOT NULL,
        PRIMARY KEY(media-id)
        FOREIGN KEY(publish-user-id) REFERENCES User(user-id) on delete cascade )
        ENGINE=INNODB;

CREATE TABLE  TV-Series-Episode (
        media-id VARCHAR(32) NOT NULL,
        episode-number INT NOT NULL,
        season-number INT NOT NULL,
        emmy-award DATE,
        PRIMARY KEY(media-id)
        FOREIGN KEY(media-id) REFERENCES Media(media-id) on delete cascade )
ENGINE=INNODB;

CREATE TABLE Movie (
        media-id VARCHAR(32) NOT NULL,
        Oscar-award DATE,
        PRIMARY KEY(media-id)
        FOREIGN KEY(media-id) REFERENCES Media(media-id) on delete cascade )
ENGINE=INNODB;

CREATE TABLE Channel(
        channel-id VARCHAR(32) NOT NULL,
        user-id VARCHAR(32) NOT NULL,
        title VARCHAR(64),
        PRIMARY KEY(channel-id)
        FOREIGN KEY(user-id) REFERENCES User(user-id) )
ENGINE=INNODB;

CREATE TABLE  CommentLike (
         user-id VARCHAR(32) NOT NULL,
         comment-id VARCHAR (32) NOT NULL,
        PRIMARY KEY( user-id, comment-id)
        FOREIGN KEY(user-id) REFERENCES User(user-id)
        FOREIGN KEY(comment-id) REFERENCES Comment(comment-id)
) ENGINE=INNODB;
```

```
CREATE TABLE  SubComment(
        parent-id VARCHAR(32) NOT NULL,
        child-id VARCHAR(32) NOT NULL,
        PRIMARY KEY(parent-id, current-id)
        FOREIGN KEY(parent-id) REFERENCES Comment(comment-id)
        FOREIGN KEY(child-id) REFERENCES Comment(comment-id)
) ENGINE=INNODB;

CREATE TABLE  GenrePreference(
        user-id VARCHAR(32) NOT NULL,
        genre-id VARCHAR(32) NOT NULL,
        PRIMARY KEY(user-id, genre-id)
        FOREIGN KEY(user-id) REFERENCES User(user-id)
        FOREIGN KEY(genre-id) REFERENCES Genre(genre-id)
) ENGINE=INNODB;

CREATE TABLE  MediaHasGenre(
        media-id VARCHAR(32) NOT NULL,
        genre-id VARCHAR(32) NOT NULL,
        PRIMARY KEY(media-id, genre-id)
        FOREIGN KEY(media-id) REFERENCES Media(media-id)
        FOREIGN KEY(genre-id) REFERENCES Genre(genre-id)
) ENGINE=INNODB;

CREATE TABLE  ChannelHasGenre(
        channel-id VARCHAR(32) NOT NULL,
        genre-id VARCHAR(32) NOT NULL,
        PRIMARY KEY(channel-id, genre-id)
        FOREIGN KEY(channel-id) REFERENCES Channel(channel-id)
        FOREIGN KEY(genre-id) REFERENCES Genre(genre-id)
) ENGINE=INNODB;

CREATE TABLE  ChannelMedia(
        media-id VARCHAR(32) NOT NULL,
        channel-id VARCHAR(32) NOT NULL,
        PRIMARY KEY(media-id, channel-id)
        FOREIGN KEY(media-id) REFERENCES Media(media-id)
        FOREIGN KEY(channel-id) REFERENCES Channel(channel-id)
) ENGINE=INNODB;
```

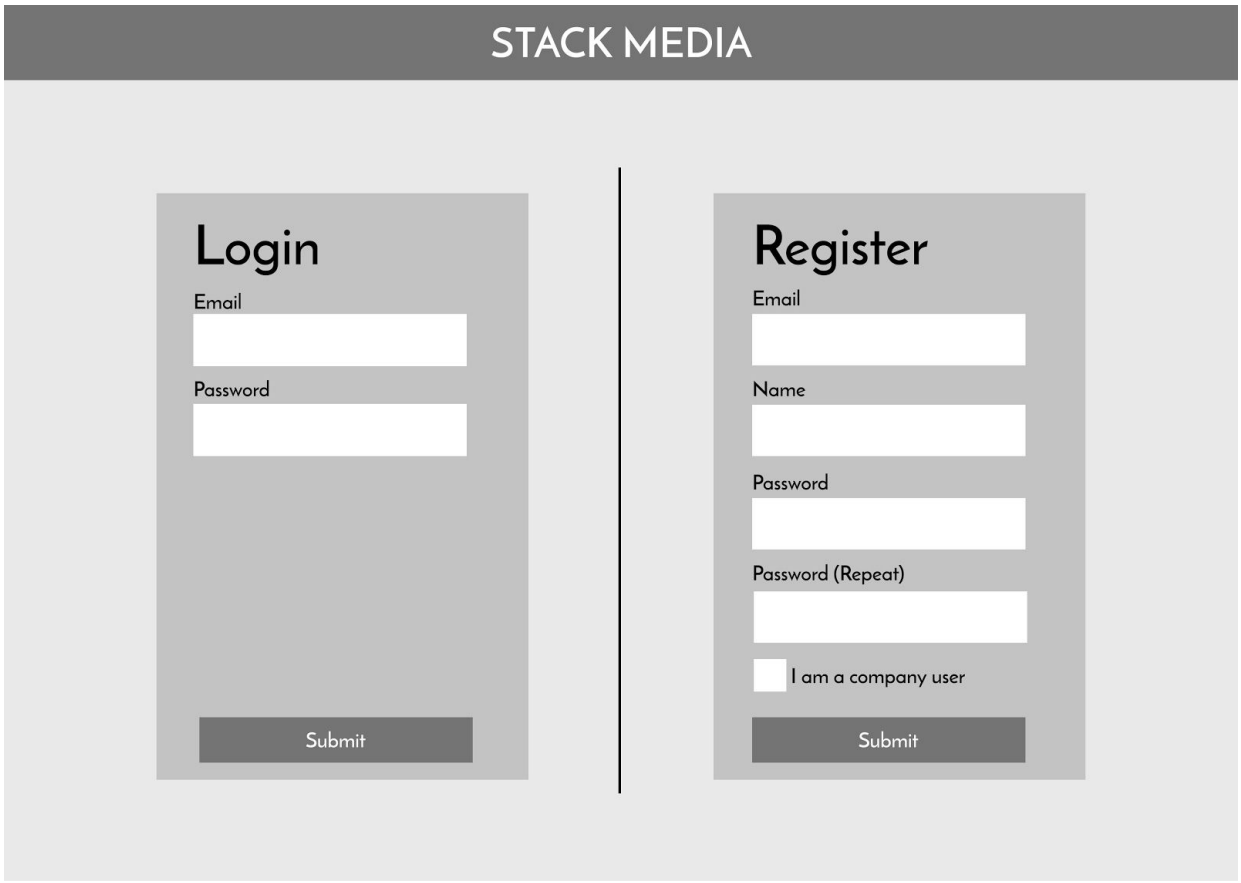
```
CREATE TABLE  Watch(
        user-id VARCHAR(32) NOT NULL,
        media-id VARCHAR(32) NOT NULL,
        Progress INT NOT NULL,
        time-stamp TIMESTAMP NOT NULL,
        PRIMARY KEY(user-id , media-id)
        FOREIGN KEY(user-id) REFERENCES User(user-id)
        FOREIGN KEY(media-id) REFERENCES Media(media-id)
) ENGINE=INNODB;

CREATE TABLE  MediaRating(
        user-id VARCHAR(32) NOT NULL,
        media-id VARCHAR(32) NOT NULL,
        rate INT NOT NULL,
        PRIMARY KEY(user-id , media-id)
        FOREIGN KEY(user-id) REFERENCES User(user-id)
        FOREIGN KEY(media-id) REFERENCES Media(media-id)
) ENGINE=INNODB;

CREATE TABLE  PartyParticipation(
        party-id VARCHAR(32) NOT NULL,
        user-id VARCHAR(32) NOT NULL,
        PRIMARY KEY(party-id, user-id)
        FOREIGN KEY(party-id) REFERENCES Party(party-id)
        FOREIGN KEY(user-id) REFERENCES User(user-id)
) ENGINE=INNODB;

CREATE TABLE  PartyInvitation(
        party-id VARCHAR(32) NOT NULL,
        user-id VARCHAR(32) NOT NULL,
        time-stamp TIMESTAMP NOT NULL,
        PRIMARY KEY(party-id, user-id)
        FOREIGN KEY(party-id) REFERENCES Party(party-id)
        FOREIGN KEY(user-id) REFERENCES User(user-id)
) ENGINE=INNODB;

CREATE TABLE  Friendship(
        friend1-id VARCHAR(32) NOT NULL,
        Friend2-id VARCHAR(32) NOT NULL,
        PRIMARY KEY(friend1-id, friend2-id)
        FOREIGN KEY(friend1-id) REFERENCES User( user-id)
        FOREIGN KEY(friend2-id) REFERENCES User( user-id)
) ENGINE=INNODB;
```

```
CREATE TABLE  FriendshipInvitation(
        Inviter-id VARCHAR(32) NOT NULL,
        Invited-id VARCHAR(32) NOT NULL,
        PRIMARY KEY(Inviter-id, Invited-id )
        FOREIGN KEY(Inviter-id) REFERENCES User( user-id)
        FOREIGN KEY(invited-id) REFERENCES User( user-id)
) ENGINE=INNODB;
```

2. UI Design and Corresponding SQL Statements

Our general UI design was inspired by Youtube, Spotify and Netflix.

2.1 Login Page UI



Login/Register Page

In this page, the user can login to the system or he/she can register by specifying his/her information along with the user type.

2.2 Login Page UI Corresponding SQL Statements

- **Login**

```
SELECT
        *
FROM
        User
WHERE
        email = emailUIText AND password = passwordUIText
```

- **Register**

First check if the user already exists in the system

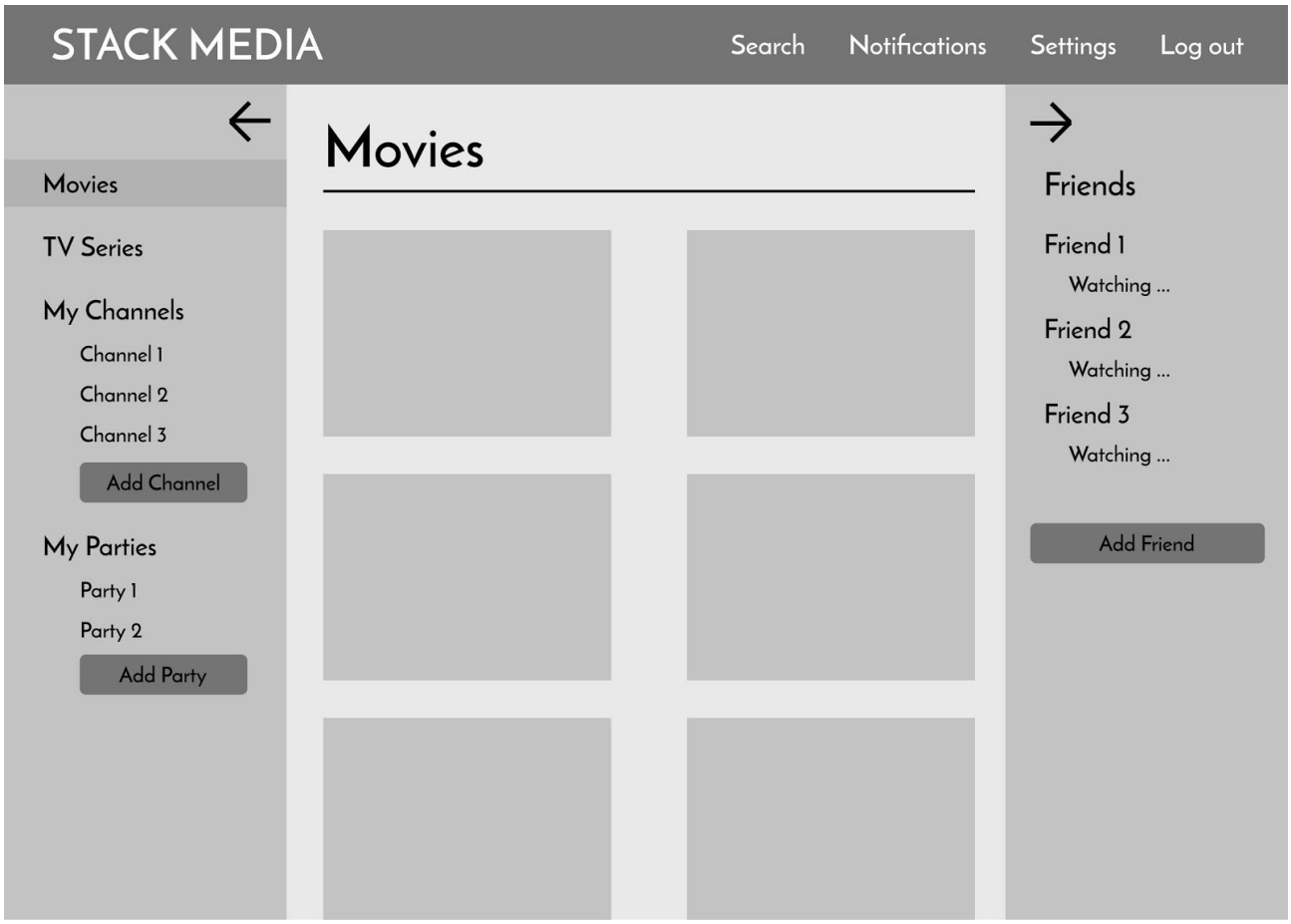
```
SELECT
        *
FROM
        User
WHERE
        email = emailUIText OR name = nameUIText
```

if the query is empty

```
INSERT INTO User
VALUES ( getNextUserID(), nameUIText, passwordUIText, isCompanyUserRadioInput)
```

getNextUserID() is server function, returns unique id.

2.3 Movies Page UI



Movies Page

In this page, movies are presented to the user according to his/her genre preferences. The user can choose a movie and start watching.

2.4 Movies Page UI Corresponding SQL Statements

- **Finding channels of the client**

```
SELECT
        title AS channelName
        channel-id
FROM
        User INNER JOIN Channel ON Channel.user-id = User.user-id
```

- **Displaying all movies based on genre preference**

```
SELECT
        M.*
FROM
        Media M INNER JOIN MediaHasGenre MHG ON M.media-id = MHG.media-id
WHERE
        MHG.genre-id in (
        SELECT
                genre-id
        FROM
                GenrePreference
        WHERE
                user-id = clientID
)
```

- **Adding new party**

```
INSERT INTO
        Party
VALUES
        (party-id, main-user-id, desired-name, desired-description, 0)
```

- **Sending friend request**

```
INSERT INTO
        FriendshipInvitation
VALUES
        (main-user-id, wanted-friend-id)
```

- **Friend activities**

```
SELECT
        F.friend2-id, M.name
FROM
        Friendship F, Watch W, Media M
```

```
WHERE
        F.friend1-id = main-user-id and Watch.user-id = F.friend2-id and
        W.media-id = M.media-id
```

- **Deleting a party**

```
DELETE FROM
        Party
WHERE
        creator-id = main-user-id and party-id = deleted-party-id
```

- **Removing a friend**

```
DELETE FROM
        Friendship
WHERE
        friend1-id = main-user-id and friend2-id = deleted-friend-id
```

- **Accepting friend request through notifications**

```
DELETE FROM
        FriendshipInvitation
WHERE
        invited-id = main-user-id and inviter-id = rejected-friend-id

INSERT INTO
        Friendship
VALUES
        (main-user-id, accepted-friend-id)
```

- **Rejecting friend request through notifications**

```
DELETE FROM
        FriendshipInvitation
WHERE
        invited-id = main-user-id and inviter-id = rejected-friend-id
```

- **Accepting party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id = user-id-variable and party-id = party-id-variable

INSERT INTO
        PartyParticipation
VALUES
        (party-id-variable, user-id,variable)
```

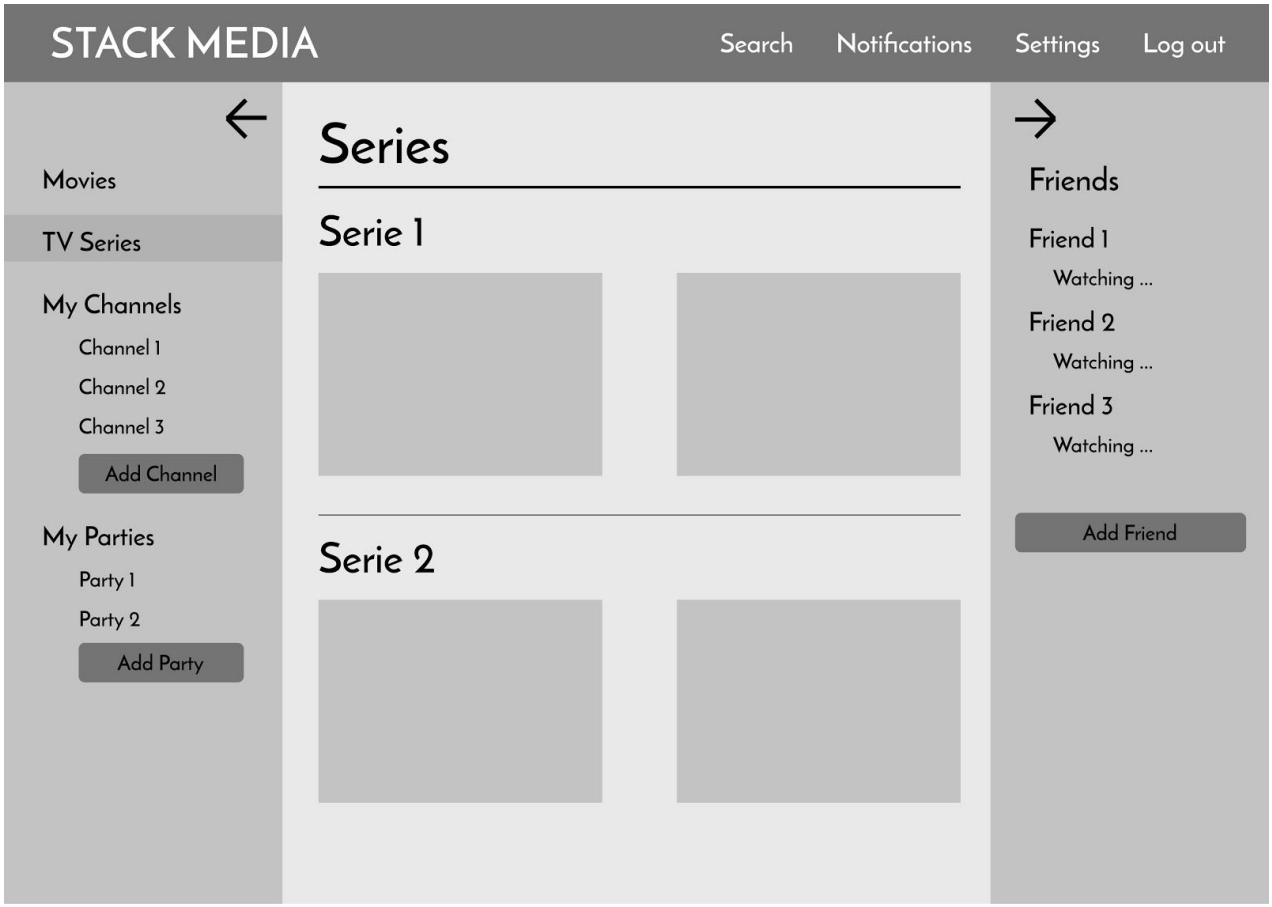
- **Rejecting party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id = user-id-variable and party-id = party-id-variable
```

- **Removing a channel**

```
DELETE FROM Channel
WHERE
        Channel.channel-id = selectedChannelID AND Channel.user-id = clientUserID
```

2.5 Series Page UI



Series Page

In this page, series are presented to the user according to his/her genre preferences. The user can choose a movie and start watching.

2.6 Series Page UI Corresponding SQL Statements

- **Finding channels of the client**

```
SELECT
        title AS channelName
        channel-id
FROM
        User INNER JOIN Channel ON Channel.user-id = User.user-id
```

- **Displaying all the tv-shows based on genre**

```
SELECT
        M.*
FROM
        Media M INNER JOIN MediaHasGenre MHG ON M.media-id = MHG.media-id
WHERE
        MHG.genre-id in (
        SELECT
                genre-id
        FROM
                GenrePreference
        WHERE
                user-id = clientID
)
```

- **Creating a new party**

```
INSERT INTO
        Party
VALUES
        (party-id, main-user-id, desired-name, desired-description, 0)
```

- **Sending a friend request**

```
INSERT INTO
        FriendshipInvitation
VALUES
        (main-user-id, wanted-friend-id)
```

- **Friend activities**

```
SELECT
        F.friend2-id, M.name
FROM
        Friendship F, Watch W, Media M
```

```
WHERE
        F.friend1-id = clientID and Watch.user-id = F.friend2-id and
        W.media-id = M.media-id
```

- **Removing a party**

```
DELETE FROM
        Party
WHERE
        creator-id = main-user-id and party-id = deleted-party-id
```

- **Removing a friend**

```
DELETE FROM
        Friendship
WHERE
        friend1-id = main-user-id and friend2-id = deleted-friend-id
```

- **Accepting a friend request through notifications**

```
DELETE FROM
        FriendshipInvitation
WHERE
        invited-id = main-user-id and inviter-id = rejected-friend-id

INSERT INTO
        Friendship
VALUES
        (main-user-id, accepted-friend-id)
```

- **Rejecting a friend request through notifications**

```
DELETE FROM
        FriendshipInvitation
WHERE
        invited-id = main-user-id and inviter-id = rejected-friend-id
```

- **Accepting a party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id = user-id-variable and party-id = party-id-variable

INSERT INTO
        PartyParticipation
VALUES
        (party-id-variable, user-id,variable)
```

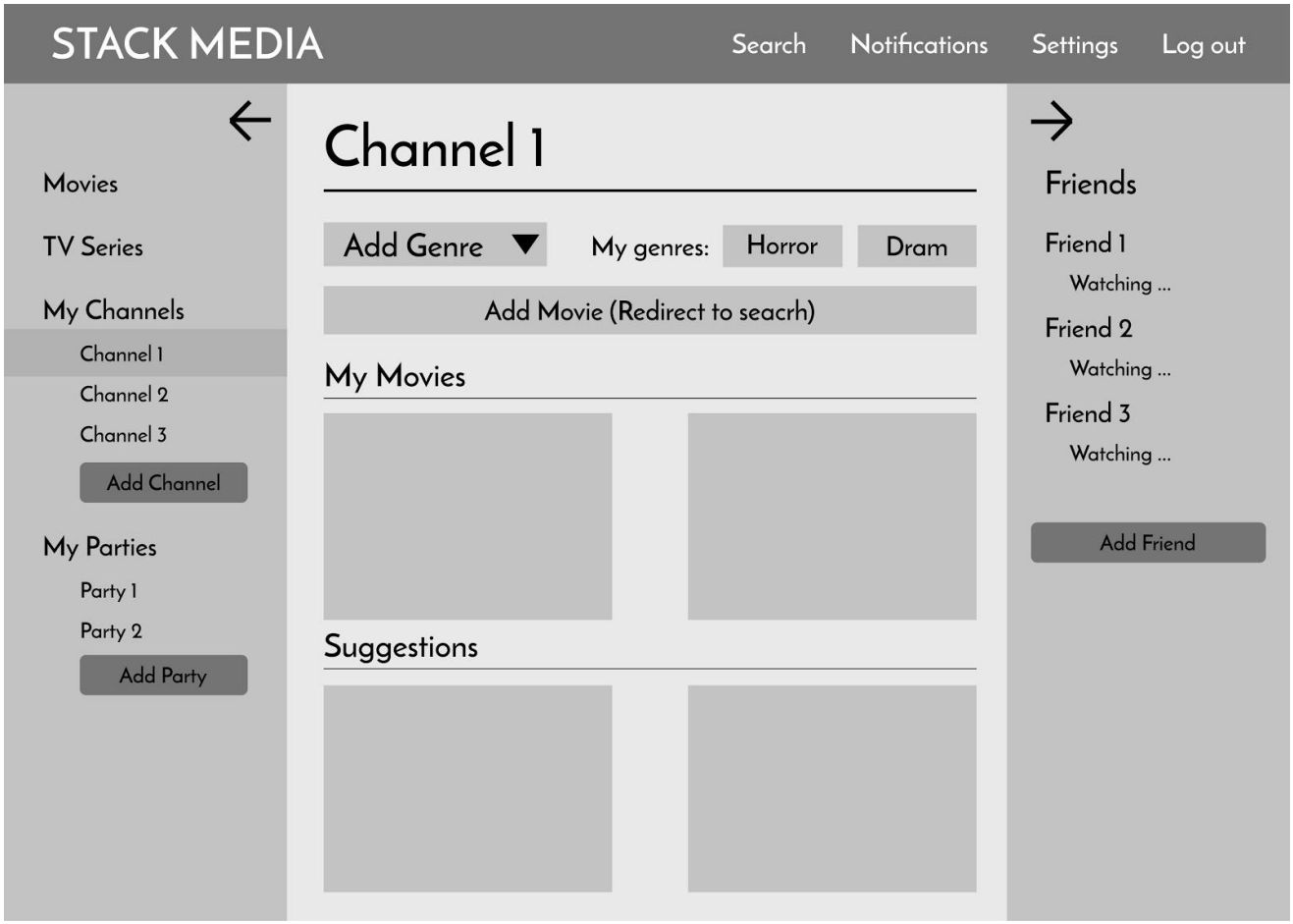
- **Rejecting a party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id = user-id-variable and party-id = party-id-variable
```

- **Removing a channel**

```
DELETE FROM Channel
WHERE
        Channel.channel-id = selectedChannelID AND Channel.user-id = clientUserID
```

2.7 Channel Page UI



Channel Page

In this page, the user can pick add/delete genres for the channel, and add movies to the channel. Also he/she can see/watch added/suggested movies. Suggestions are handled by the system according to the selected genres.

2.8 Channel Page UI Corresponding SQL Statements

- **Finding channels of the client**

```
SELECT
        title AS channelName
        channel-id
FROM
        User INNER JOIN Channel ON Channel.user-id = User.user-id
```

- **Finding every movie of a channel that belongs to the client**

```
SELECT
        Media.*
FROM
        Media INNER JOIN ChannelMedia ON Media.media-id = ChannelMedia.media-id
WHERE
        ChannelMedia.channel-id = curChannelID
```

(curChannelID is the channel selected on the menu)

- **Creating a new party**

```
INSERT INTO
        Party
VALUES
        (party-id, main-user-id, desired-name, desired-description, 0)
```

- **Sending a friend request**

```
INSERT INTO
        FriendshipInvitation
VALUES
        (main-user-id, wanted-friend-id)
```

- **Friend activities**

```
SELECT
        F.friend2-id, M.name
FROM
        Friendship F, Watch W, Media M
WHERE
        F.friend1-id = main-user-id and Watch.user-id = F.friend2-id and
        W.media-id = M.media-id
```

- **Removing a party**

```
DELETE FROM
```

```
    Party
WHERE
    creator-id = main-user-id and party-id = deleted-party-id
```

- **Removing a friend**

```
DELETE FROM
    Friendship
WHERE
    friend1-id = main-user-id and friend2-id = deleted-friend-id
```

- **Accepting a friend request through notifications**

```
DELETE FROM
    FriendshipInvitation
WHERE
    invited-id = main-user-id and inviter-id = rejected-friend-id

INSERT INTO
    Friendship
VALUES
    (main-user-id, accepted-friend-id)
```

- **Rejecting a friend request through notifications**

```
DELETE FROM
    FriendshipInvitation
WHERE
    invited-id = main-user-id and inviter-id = rejected-friend-id
```

- **Accepting a party invitation through notifications**

```
DELETE FROM
    PartyInvitation
WHERE
    user-id = user-id-variable and party-id = party-id-variable

INSERT INTO
    PartyParticipation
VALUES
    (party-id-variable, user-id,variable)
```

- **Rejecting a party invitation through notifications**

```
DELETE FROM
    PartyInvitation
WHERE
    user-id = user-id-variable and party-id = party-id-variable
```

- **Removing a channel**

```
DELETE FROM Channel
WHERE
        Channel.channel-id = selectedChannelID AND Channel.user-id = clientUserID
```

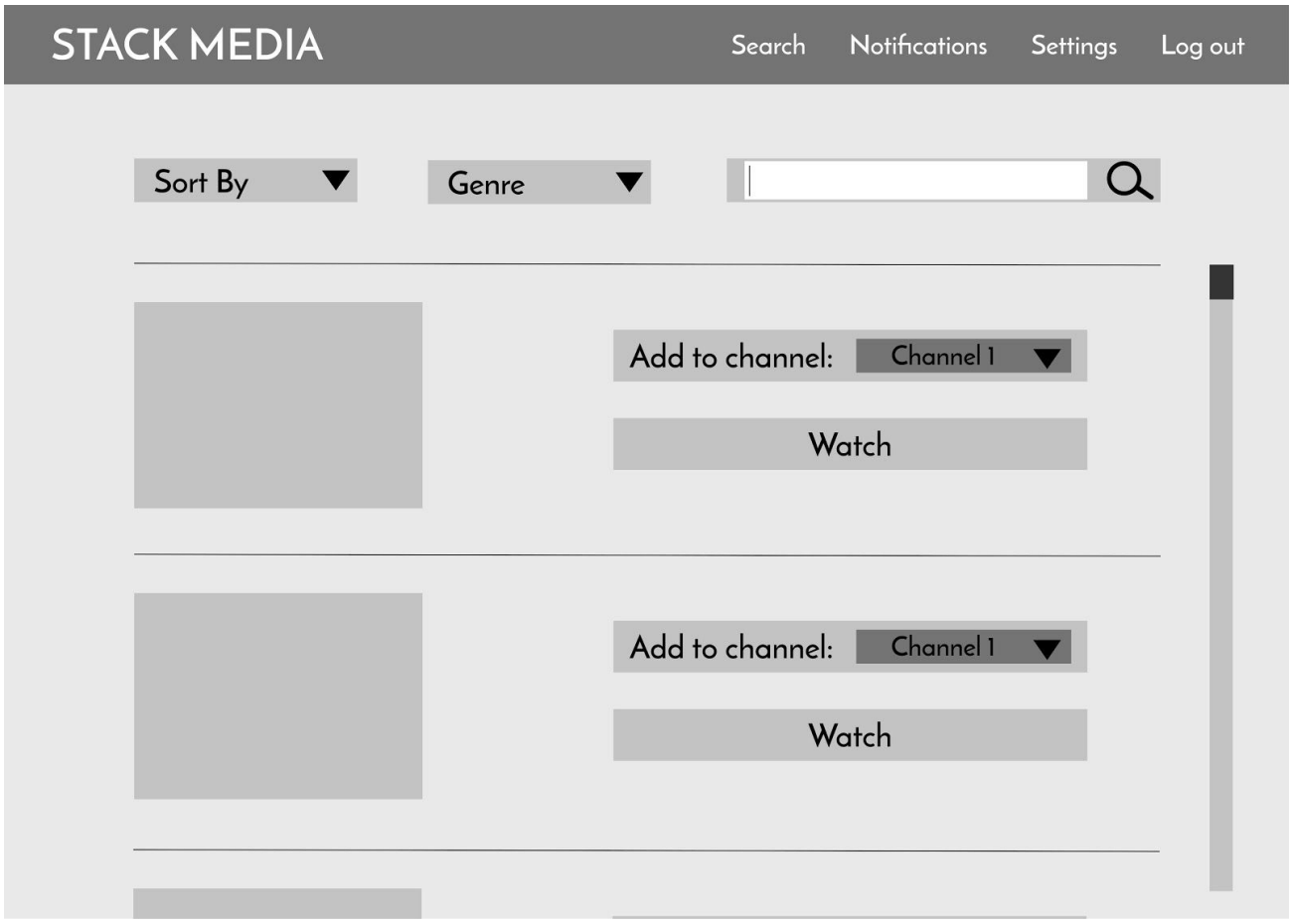
- **Finding suggested media based on channel genre**

```
SELECT
        M.*
FROM
        Media M INNER JOIN MediaHasGenre MHG ON MHG.media-id = M.media-id
WHERE
        MHG.genre-id in (
        SELECT
                genre-id
        FROM
                ChannelHasGenre
        WHERE
                channel-id = clientCurChannelID
)
```

- **Editing genre of a channel**

```
//deletion
DELETE FROM ChannelHasGenre
WHERE
        channel-id = curChannelID AND genre-id = genreIDToDelete

// to insert new genre
INSERT INTO ChannelHasGenre
VALUES( curChannelID, genreIDToAdd)
```

2.9 Search Page UI



Search Page - Standard User

In this page, the user can search for movies by specifying genre and a media name. Also he/she can sort the results. Next, he/she can add the media to channels or directly watch.

2.10 Search Page UI Corresponding SQL Statements

- **Searching media**

```
SELECT
        *
FROM
        Media M
WHERE
        DIFFERENCE( name, movieNameUIText) = 4 AND
        genreUISELECTION in (
                SELECT
                        Genre.title
```

```
            FROM
                HasGenre INNER JOIN Genre ON HasGenre.genre-id = Genre.genre-id
            WHERE
                M.media-id = HasGenre.media-id
    )
ORDER BY
    sortUISelection DESC
```

- **Adding media to a channel after searching**

```
INSERT INTO ChannelMedia
VALUES
    ( media-id,
    channel-id );
```

- **Getting every available genre**

```
SELECT
    genre-id,
    title
FROM
    Genre
```

- **Accepting a friend request through notifications**

```
DELETE FROM
    FriendshipInvitation
WHERE
    invited-id = main-user-id and inviter-id = rejected-friend-id

INSERT INTO
    Friendship
VALUES
    (main-user-id, accepted-friend-id)
```

- **Rejecting a friend request through notifications**

```
DELETE FROM
    FriendshipInvitation
WHERE
    invited-id = main-user-id and inviter-id = rejected-friend-id
```

- **Accepting a party invitation through notifications**

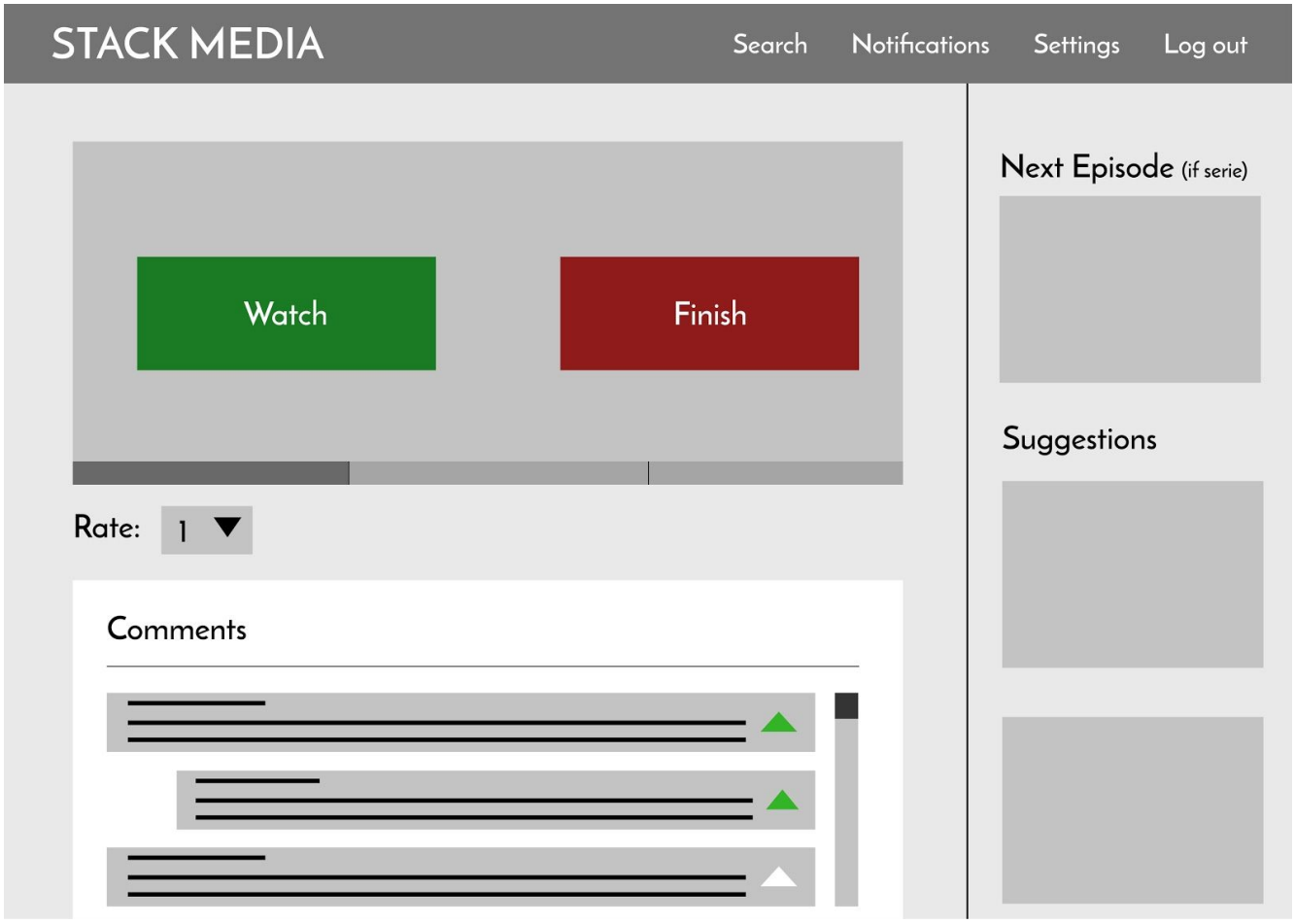
```
DELETE FROM
    PartyInvitation
WHERE
    user-id = user-id-variable and party-id = party-id-variable
```

```
INSERT INTO
        PartyParticipation
VALUES
        (party-id-variable, user-id,variable)
```

- **Rejecting a party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id = user-id-variable and party-id = party-id-variable
```

2.11 Watch Page UI



Watch Page

In this page, the user can watch a media by using the watch and finish buttons, also he/she can rate the media and leave a comment. At the right side, the system suggests the user to watch similar media and presents the next episode if this is a series.

2.12 Watch Page UI Corresponding SQL Statements

- **Accepting a friend request through notifications**

```
DELETE FROM
        FriendshipInvitation
WHERE
        invited-id = main-user-id and inviter-id = rejected-friend-id

INSERT INTO
        Friendship
VALUES
        (main-user-id, accepted-friend-id)
```

- **Rejecting a friend request through notifications**

```
DELETE FROM
        FriendshipInvitation
WHERE
        invited-id = main-user-id and inviter-id = rejected-friend-id
```

- **Accepting a party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id = user-id-variable and party-id = party-id-variable

INSERT INTO
        PartyParticipation
VALUES
        (party-id-variable, user-id,variable)
```

- **Rejecting a party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id = user-id-variable and party-id = party-id-variable
```

- **Rating of the media in display**

```
SELECT
        M.media-id
        AVG(rate)
FROM
```

```
        Media M INNER JOIN MediaRating ON M.media-id = MediaRating.media-id
GROUP BY
        M.media-id
```

- **Comments belonging to the media in display**

```
SELECT
        C.*
FROM
        Comment C
WHERE
        media-id = curMediaID
ORDER BY
        time-stamp DESC
```

- **Comments of a comment**

(by DB design, we only allow a comment to have only a single parent as of in Youtube. This is a design limitation we have made. We do think that a Reddit style of comment threads are unreadable and messy.)

```
SELECT
        C.*
FROM
        Comment C
WHERE
        comment-id in (
        SELECT
                child-id
        FROM
                SubComment
        WHERE
                parent-id = selectedCommentID
)
```

- **Media suggestions based on media genre that is in display**

```
SELECT
        M.name
FROM
        GenrePreference GP, HasGenre HG, Media M
WHERE
        GP.user-id = main-user-id and GP.genre-id = HG.genre-id and
        HG.media-id = M.media-id
```

- **Leaving a rating to the media in display**

Check if user has rating

```
SELECT
        *
FROM
        MediaRating
WHERE
        Media-id = curMediaID AND user-id = clientID
```

If there is not a rating by the user before

```
INSERT INTO MediaRating
VALUES
        ( clientUserID,
        curMovieID,
        selectedRating );
```

If there is a rating by the client

```
UPDATE MediaRating
SET rating = selectedRating
WHERE
        media-id = curMediaID AND user-id = clientID
```

- **Getting watch progress of a client for the media in display**

```
SELECT
        progress
FROM
        Watch
WHERE
        media-id = curMediaID AND user-id = clientUserID
```

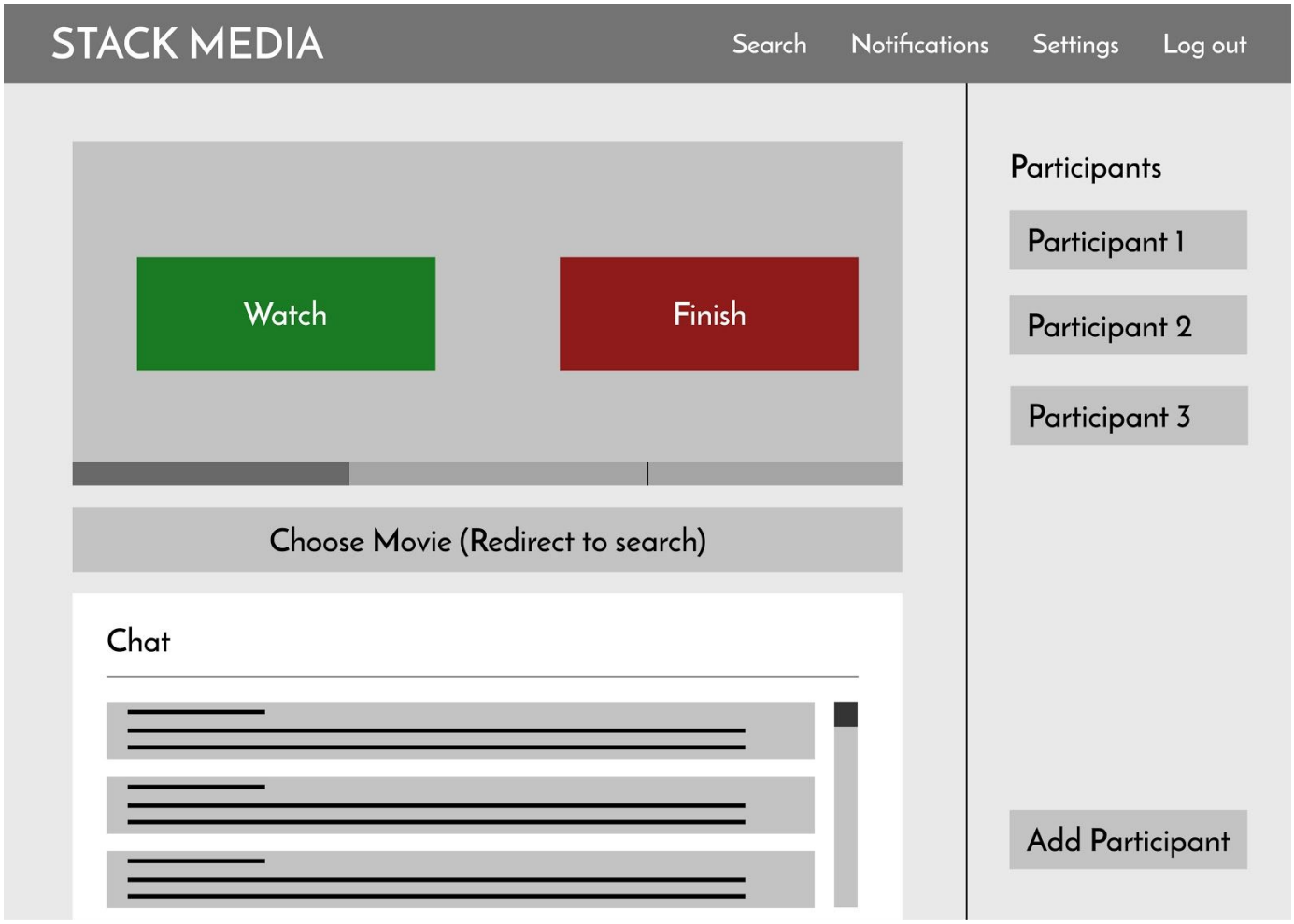
- **Watch button press to simulate watch action**

```
UPDATE Watch
SET progress = cachedProgress + 1, time-stamp = TIMESTAMP()
WHERE
        Media-id = curMediaID AND user-id = clientUserID
```

- **Finish button press to simulate finish action**

```
UPDATE Watch
SET progress = cachedProgress + 1, time-stamp = TIMESTAMP()
WHERE
        Media-id = curMediaID AND user-id = clientUserID
```

2.13 Party Page UI



Party Page

In this page, the creator of the party can select a movie and make the participants watch by using watch and finish buttons. The creator can also add/delete participants to the party. At the bottom, there is a chat for all participants.

2.14 Party Page UI Corresponding SQL Statements

- **Adding a participant to the party**

```
INSERT INTO
        PartyParticipation
VALUES
        (party-id-variable, user-id,variable)
```

- **Removing a participant from the party**

```
DELETE FROM
        PartyParticipation
WHERE
        user-id = user-id-variable and party-id = party-id-variable
```

- **Accepting a friend request through notifications**

```
DELETE FROM
        FriendshipInvitation
WHERE
        invited-id = main-user-id and inviter-id = rejected-friend-id

INSERT INTO
        Friendship
VALUES
        (main-user-id, accepted-friend-id)
```

- **Rejecting a friend request through notifications**

```
DELETE FROM
        FriendshipInvitation
WHERE
        invited-id = main-user-id and inviter-id = rejected-friend-id
```

- **Accepting a party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id == user-id-variable and party-id == party-id-variable

INSERT INTO
        PartyParticipation
VALUES
        (party-id-variable, user-id,variable)
```

- **Rejecting a party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id == user-id-variable and party-id == party-id-variable
```

- **Getting watch progress for the current client**

```
SELECT
```

```
        progress
FROM
        Watch
WHERE
        media-id = curMediaID AND user-id = clientUserID
```

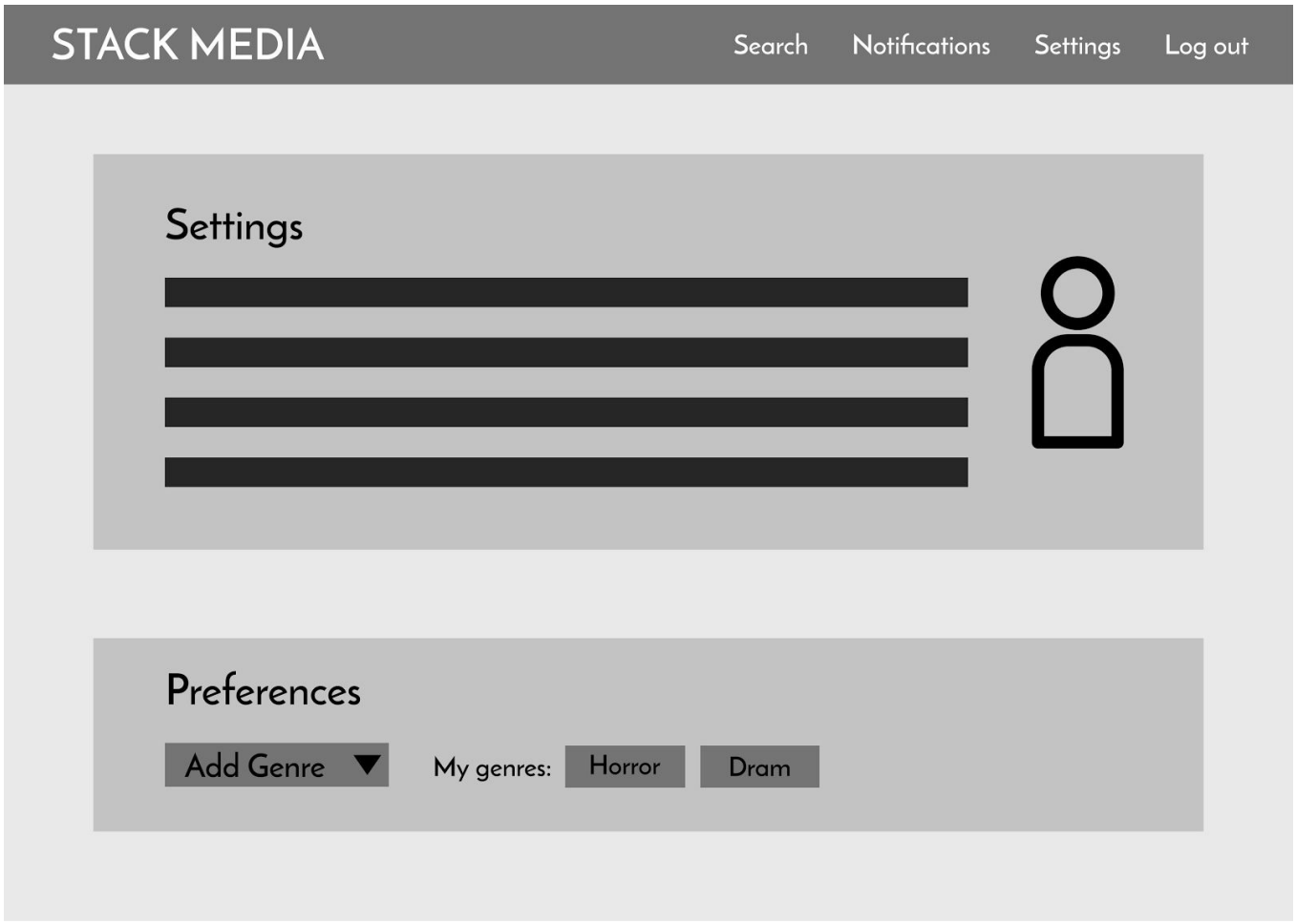
- **Watch button press to simulate watch action**

```
UPDATE Watch
SET progress = cachedProgress + 1, time-stamp = TIMESTAMP()
WHERE
        Media-id = curMediaID AND user-id = clientUserID
```

- **Finish button press to simulate finish action**

```
UPDATE Watch
SET progress = cachedProgress + 1, time-stamp = TIMESTAMP()
WHERE
        Media-id = curMediaID AND user-id = clientUserID
```

2.15 Settings Page UI



Settings Page

In this page, the user can see/change his/her settings, and also add/delete genres for specialized suggestions.

2.16 Settings Page UI Corresponding SQL Statements

- **Accepting a friend request through notifications**

```
DELETE FROM
        FriendshipInvitation
WHERE
        invited-id = main-user-id and inviter-id = rejected-friend-id

INSERT INTO
        Friendship
```

```
VALUES
        (main-user-id, accepted-friend-id)
```

- **Refusing a friend request through notifications**

```
DELETE FROM
        FriendshipInvitation
WHERE
        invited-id = main-user-id and inviter-id = rejected-friend-id
```

- **Accepting a party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id = user-id-variable and party-id = party-id-variable

INSERT INTO
        PartyParticipation
VALUES
        (party-id-variable, user-id,variable)
```

- **Refusing a party invitation through notifications**

```
DELETE FROM
        PartyInvitation
WHERE
        user-id = user-id-variable and party-id = party-id-variable
```

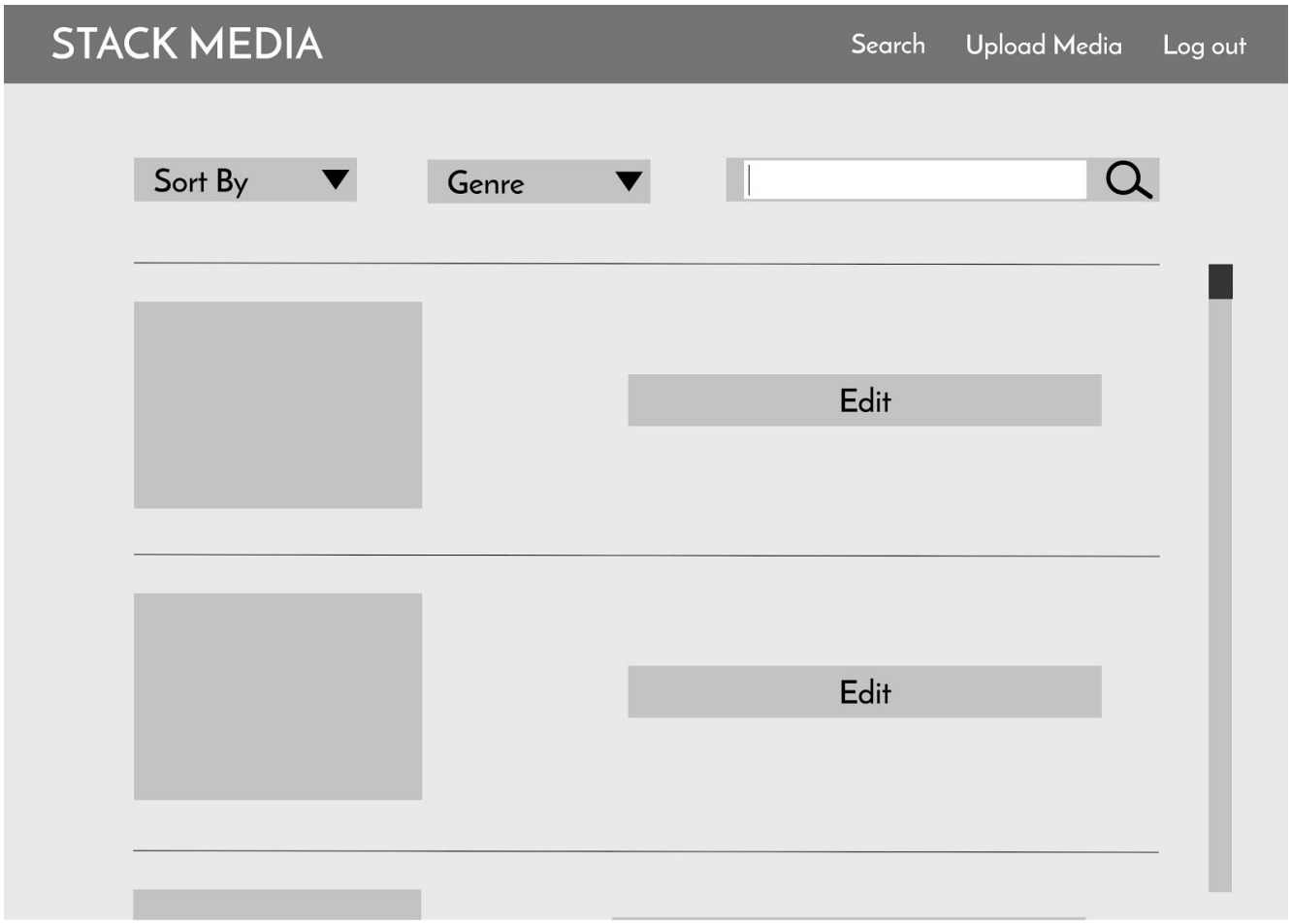
- **Adding genre preference for suggestions**

```
INSERT INTO GenrePreference
VALUES
        ( curUserID,
        genreID );
```

- **Removing a genre preference for suggestions**

```
DELETE FROM GenrePreference
WHERE
        user-id = clientUserID AND genre-id = selectedGenreID
```

2.17 Search Page Company UI



Search Page - Company User

In this page, the user can search for movies by specifying genre and a movie name. Also he/she can sort the results. Next, he/she can edit the media (the button redirects to Upload/Edit Media Page).

2.18 Search Page Company UI Corresponding SQL Statements

- **Searching media**

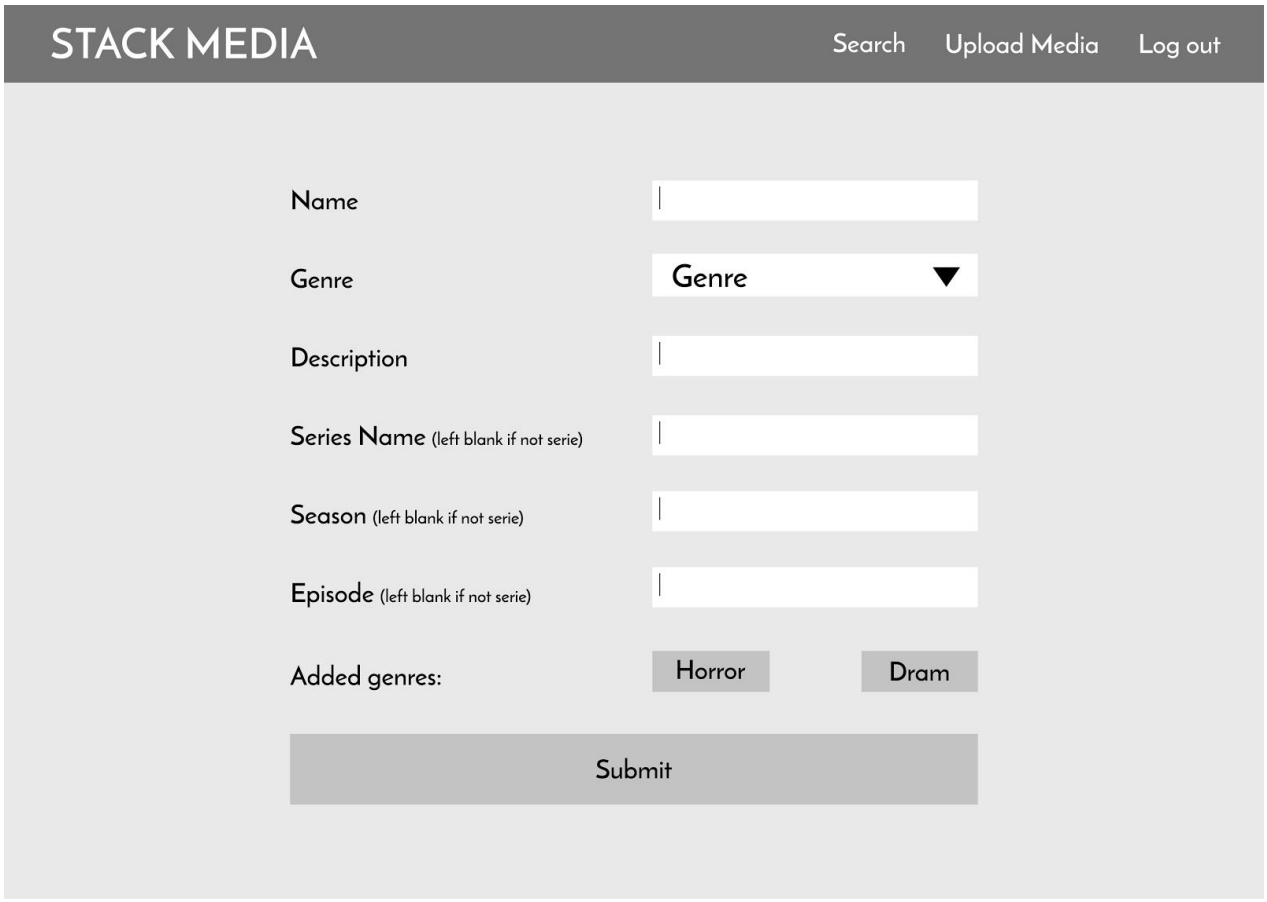
```
SELECT
        *
FROM
        Media M
WHERE
        DIFFERENCE( name, movieNameUIText) = 4 AND
        genreUISELECTION in (
                SELECT
```

```
            Genre.title
        FROM
            HasGenre INNER JOIN Genre ON HasGenre.genre-id = Genre.genre-id
        WHERE
            M.media-id = HasGenre.media-id
    )
ORDER BY
    sortUISelection DESC
```

- **Editing an existing media**

```
// updating
UPDATE Media
SET
    name = nameArg,
    description =descArg,
    path = mediaLinkArg,
    duration = durationArg
WHERE
    Media.media-id = selectedMediaID

// deleting
DELETE FROM Media
WHERE
    Media.media-id = selectedMediaID
```

2.19 Upload/Edit Media Company Page UI



Upload/Edit Media Page

The company user can upload a new media by specifying its attributes, and also he/she can edit the attributes of a selected media (the edit button in the search page redirects here for edit).

2.20 Upload/Edit Media Company Page UI Corresponding SQL Statements

- **Uploading a media**

movie

```
INSERT INTO Media
VALUES
        ( getUniqueMediaID(),
        curUserID,
        NULL,
        mediaNameUIText,
        descriptionUIText,
        pathUIText,
```

durationUIText,
TIMESTAMP());

INSERT INTO Movie
VALUES
(movieID, NULL);
movieID is cached on the server.

tv-show
INSERT INTO Media
VALUES
(getUniqueMediaID(),
curUserID,
seriesUINameText,
mediaNameUIText,
descriptionUIText,
pathUIText,
durationUIText,
TIMESTAMP());

INSERT INTO TV-Series-Episode
VALUES
(mediaID,
episodeUIText,
seasonUIText,
NULL);