University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: Aug 5, 2024

Group Number: 22

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Suhayl Patel	74030826	z8q5f	suhayl.patel@outlook.com
Keean Vidyarthi	80912504	k9x4u	keeanvid1@gmail.com
Weena Wibowo	74581323	r8q5n	weenawibowo@hotmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

GITHUB REPO LINK:
https://github.students.cs.ubc.ca/CPSC304-2024S-T2/project_k9x4u_r8q5n_z8q5f/

SQL SCRIPT LINK FROM GITHUB REPO:
https://github.students.cs.ubc.ca/CPSC304-2024S-T2/project_k9x4u_r8q5n_z8q5f/blob/main/main.sql/

Project Description

Our project is a representation of the structure of a streaming service, from the perspective of a streaming service provider (ex. Netflix). We've decided to model entities like Users, Reviews, Media, and additional aspects that would be beneficial for a streaming service provider to manage.

This application is designed for streaming service providers to efficiently manage and monitor various aspects of their service to create a comprehensive and user-friendly database solution.

Our application is capable of sorting through the database to find media based on user-selected parameters, such as by Genre or Media Producer, and tracking which user-written reviews are associated with which piece of media. This provides insight to the streaming service provider about how Users interact with Media and will allow a streaming service provider to more effectively manage their operations and understand their user base by optimizing content delivery.

Our Schema Changes include:

"Review" is now only a weak entity of "User", and has a many-to-one relationship "Gets" with "Media." Previously, we structured "Review" to be a weak entity of both "User" and "Media," which was incorrect as a weak entity relationship should only involve two entities.

The following schema changes are due to syntax errors as the names we used previously use characters and keywords that are not permitted for use in SQL:

"In" relationship between "Media" and "Actor" renamed to "Acts"

“User” entity renamed to “UserAccount”

“phone#” attribute of “User” renamed to “phoneNum”

“star#” attribute of “Review” renamed to “starNum”

“episode#” attribute of “Episode” renamed to “episodeNum”

“#ofEpisodes” attribute of “TV Show” renamed to “numEpisodes”

List of Relations: (NEED THIS FOR ALL TABLES LISTED IN MAIN.SQL, UNDERLINE PK BOLD FK)

- UserAccount(email, phoneNum, userName, dateOfBirth)

`UserAccount`

email	phoneNum	userName	dateOfBirth
bob@gmail.com	2041238888	BOB ROSS	1995-09-12
jane@gmail.com	4038972345	JANE DOE	1990-06-08
ruth@hotmail.com	6045892654	RUTH THEBOSS	2000-01-10
michael@hotmail.com	5876231890	MICHAEL GODAD	2002-05-09
john@gmail.com	7781459898	JOHN DONKINS	2004-03-04

- Media(mediaID, mediaTitle, **producerID**, **genreName**)

`Media`

mediaID	mediaTitle	producerID	genreName
mid1	Great Movie 1	mpid1	Horror
mid2	Spectacular Movie 2	mpid2	Comedy Adventure
mid3	OK Movie 3	mpid3	Action
mid4	OK Movie 5	mpid4	Romance
mid5	Decent Movie 3	mpid5	Thriller
mid6	Great TV show 1	mpid1	Horror
mid7	Spectacular TV show	mpid2	Romance
mid8	Mid TV show	mpid3	Fantasy
mid9	OK TV show 1	mpid4	Fiction
mid10	Decent TV show 2	mpid5	Western

- Director(directorID, directorName)

`Director`

directorID	directorName
did1	Christopher Nolan
did2	Steven Spielberg
did3	Martin Scorsese
did4	James Cameron
did5	Michael Bay

- Actor(actorID, actorName)

`Actor`

actorID	actorName
aid1	Tom Holland
aid2	Weena Wibowo
aid3	Suhayl Patel
aid4	Keean Vidyarthi
aid5	John Pork

- MediaProducer(producerID, producerName)

`MediaProducer`

producerID	producerName
mpid1	Walt Disney
mpid2	Pixar
mpid3	Paramount
mpid4	Netflix
mpid5	Universal Studio

- Genre2(numOfMedia, isPopularGenre)

`Genre2`

numOfMedia	isPopularGenre
0	0
1	0
...	...
100	1

- Genre1(genreName, **numOfMedia**)

`Genre1`

genreName	numOfMedia
Horror	43
Romance Sci-Fi	2
Thriller	92
Action	77
Comedy Adventure	15
...	...

- Subscription1(subscriptionID, **duration**, **email**, startDate, endDate)

`Subscription1`

subscriptionID	duration	email	startDate	endDate
sid1	ANNUALLY	bob@gmail.com	2021-11-27	2022-11-27
sid2	MONTHLY	jane@gmail.com	2023-04-01	2023-05-01
sid3	DAILY	ruth@hotmail.com	2023-11-03	2023-12-03
sid4	WEEKLY	michael@hotmail.com	2023-02-15	2024-02-22
sid5	BIWEEKLY	john@gmail.com	2024-01-01	2024-01-14

- Subscription2(duration, price)

`Subscription2`

duration	price
ANNUALLY	100
MONTHLY	10
DAILY	4
WEEKLY	6
BIWEEKLY	8

- Movie(**mediaID**, duration)

`Movie`

mediaID	duration
mid1	124
mid2	189
mid3	143
mid4	93
mid5	67

- TVShow(**mediaID**, numOfEpisodes)

`TVShow`

mediaID	numOfEpisodes
mid6	14
mid7	20
mid8	57
mid9	4
mid10	25

- Consumes(**mediaID, email**)

`Consumes`

mediaID	email
mid1	bob@gmail.com
mid2	jane@gmail.com
mid3	ruth@hotmail.com
mid4	michael@hotmail.com
mid5	john@gmail.com

- Acts(**mediaID, actorID**)

`Acts`

mediaID	actorID
mid1	aid1
mid2	aid2
mid3	aid3
mid4	aid4
mid5	aid5

- Review_Gets_Writes(reviewID, commentText, starNum, **email, mediaID**)

`Review_Gets_Writes`

reviewID	commentText	starNum	email	mediaID
rid1	This movie sucked!	1	bob@gmail.com	mid1
rid2	I am a new person after watching this. RECOMMEND!!!!	5	jane@gmail.com	mid2
rid3	This movie was great!	3	ruth@hotmail.com	mid3
rid4	This show was so epic!	4	michael@hotmail.com	mid4

- Episode_Has(**mediaID**, episodeNum, episodeName, season)

`Episode_Has`

mediaID	episodeNum	episodeName	season
mid6	12	Trifling tree tops	1
mid7	1	Pilot	3
mid8	6	Finale	7
mid9	2	Roundness	2
mid10	11	Uh Oh!	3

- Likes(**email**, **genreName**)

`Likes`

email	genreName
bob@gmail.com	Horror
jane@gmail.com	Comedy Adventure
ruth@hotmail.com	Action
michael@hotmail.com	Romance
john@gmail.com	Thriller

SQL Queries:

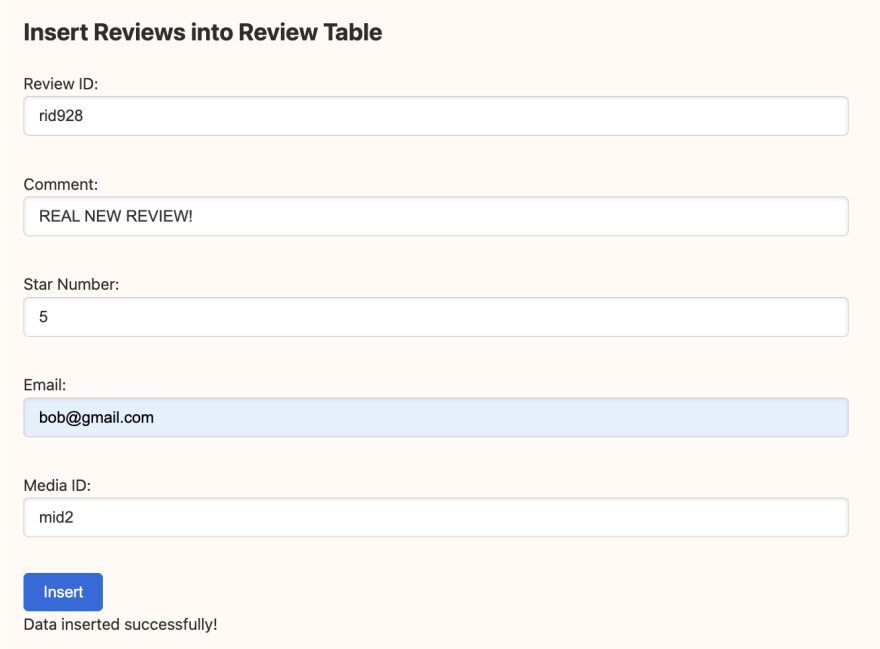
- **Insert** (Line 115 in appService.js):

```
async function insertReview(reviewID, commentText, starNum, email,

mediaID) {
```

```
    return await withOracleDB(async (connection) => {
        const result = await connection.execute(
            `INSERT INTO Review_Gets_Writes (reviewID, commentText,
starNum, email, mediaID) VALUES (:reviewID, :commentText, :starNum,
:email, :mediaID)`,
            [reviewID, commentText, starNum, email, mediaID],
            { autoCommit: true }


        );
        console.log(reviewID);
        return result.rowsAffected && result.rowsAffected > 0;
    }).catch(() => {
        return false;
    });
}
```



rid928	REAL NEW REVIEW!	5	bob@gmail.com	mid2

- **Delete** (Line 131 in appService.js):

```
async function deleteReview(reviewID, email) {
    return await withOracleDB(async (connection) => {
        const result = await connection.execute(
            `DELETE FROM Review_Gets_Writes WHERE reviewID = :reviewID
AND email = :email`,
            [reviewID, email],
            { autoCommit: true }
        );

        return result.rowsAffected && result.rowsAffected > 0;
    }).catch((err) => {
        console.error(err);
        return false;
    });
}
```

- **Update** (Line 162 in appService.js):

```
async function updateUserPhone(oldPhone, newPhone) {
    return await withOracleDB(async (connection) => {
        const result = await connection.execute(
            `UPDATE UserAccount SET phoneNum = :newPhone WHERE phoneNum
```

```
= :oldPhone`,
            [oldPhone, newPhone],
            { autoCommit: true }
        );

        return result.rowsAffected && result.rowsAffected > 0;
    }).catch((err) => {
        console.error(err);
        return false;
    });
}
```

Show User Table

Email	Phone #	Full Name	Date of birth
bob@gmail.com	6047382020	BOB ROSS	1995-09-12
jane@gmail.com	40389723293874	JANE DOE	1990-06-08
ruth@hotmail.com	6045892654	RUTH THEBOSS	2000-01-10
michael@hotmail.com	5876231890	MICHAEL GODAD	2002-05-09
john@gmail.com	2222	JOHN DONKINS	2004-03-04

Update Phone Number in User Table

The values are case sensitive and if you enter in the wrong case, the update statement will not do anything.

Old Phone Number:

2222

New Phone Number:

12345

Update

Show User Table

Email	Phone #	Full Name	Date of birth
bob@gmail.com	6047382020	BOB ROSS	1995-09-12
jane@gmail.com	40389723293874	JANE DOE	1990-06-08
ruth@hotmail.com	6045892654	RUTH THEBOSS	2000-01-10
michael@hotmail.com	5876231890	MICHAEL GODAD	2002-05-09
john@gmail.com	12345	JOHN DONKINS	2004-03-04

Update Phone Number in User Table

The values are case sensitive and if you enter in the wrong case, the update statement will not do anything.

Old Phone Number:

2222

New Phone Number:

12345

Update

Phone updated successfully!

- **Selection** (Line 198 in appService.js):

```
async function selectReview() {
   return await withOracleDB(async (connection) => {
       const result = await connection.execute(
           `SELECT * FROM Review_Gets_Writes WHERE starNum > 3`
       );
```

```
        return result.rows;
    }).catch(() => {
        return [];
    });
}
```

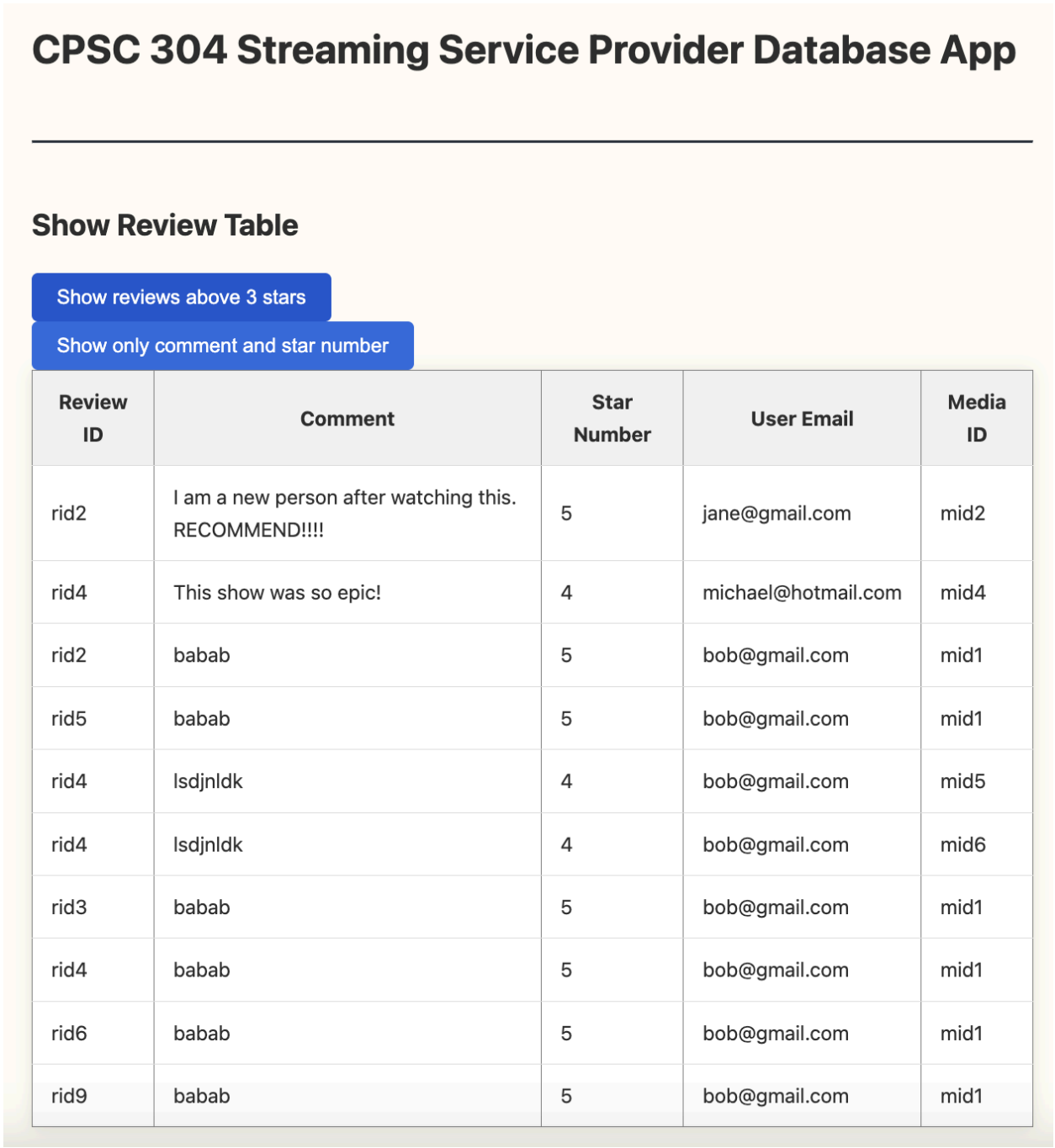
CPSC 304 Streaming Service Provider Database App

Show Review Table

Show reviews above 3 stars

Show only comment and star number

Review ID	Comment	Star Number	User Email	Media ID
rid1	skdjlsd	2	bob@gmail.com	mid8
rid5	THIS IS A NEW REVIEW	3	bob@gmail.com	mid7
rid1	This movie sucked!	1	bob@gmail.com	mid1
rid2	I am a new person after watching this. RECOMMEND!!!!	5	jane@gmail.com	mid2
rid3	This movie was great!	3	ruth@hotmail.com	mid3
rid4	This show was so epic!	4	michael@hotmail.com	mid4
rid5	Mid TV show to be honest	2	john@gmail.com	mid5
rid2	www	2	bob@gmail.com	mid3
rid2	babab	5	bob@gmail.com	mid1
rid5	babab	5	bob@gmail.com	mid1



Review ID	Comment	Star Number	User Email	Media ID
rid2	I am a new person after watching this. RECOMMEND!!!!	5	jane@gmail.com	mid2
rid4	This show was so epic!	4	michael@hotmail.com	mid4
rid2	babab	5	bob@gmail.com	mid1
rid5	babab	5	bob@gmail.com	mid1
rid4	lsdjnldk	4	bob@gmail.com	mid5
rid4	lsdjnldk	4	bob@gmail.com	mid6
rid3	babab	5	bob@gmail.com	mid1
rid4	babab	5	bob@gmail.com	mid1
rid6	babab	5	bob@gmail.com	mid1
rid9	babab	5	bob@gmail.com	mid1

- **Projection** (Line 211 in appService.js):

```
async function projectReview() {
   return await withOracleDB(async (connection) => {
       const result = await connection.execute(
           `SELECT commentText,starNum FROM Review`
```

```
        );

        return result.rows;

    }).catch(() => {

        return [];

    });
}
```

- **Join** (Line 222 in appService.js):

```
async function joinMediaProducer(producerID) {

    return await withOracleDB(async (connection) => {

        const result = await connection.execute(

            `SELECT mediaID

              FROM Media m, MediaProducer p

              WHERE m.ProducerID = p.ProducerID`,

            [producerID],

            { autoCommit: true }

        );


        return result.rows;

    }).catch(() => {

        return false;

    });
}
```

- **Aggregation with Group By** (Line 255 in appService.js):

```
async function findMediaGenre(genreName) {

    return await withOracleDB(async (connection) => {

        const result = await connection.execute(
```

```
            `SELECT mediaTitle
                FROM Media m, Genre g
                WHERE m.genreName = g.genreName
                GROUP BY m.genreName`,
            [genreName],
            { autoCommit: true }
        );

        return result.rows;
    }).catch(() => {
        return false;
    });
}
```

- **Aggregation with Having** (Line 238 in appService.js):

```
async function findReviewedMedia() {
    return await withOracleDB(async (connection) => {
        const result = await connection.execute(
            `SELECT  mediaTitle
                FROM Media m, Review r
                WHERE m.mediaID = r.mediaID
                GROUP BY m.mediaTitle
                HAVING COUNT(*) > 5`

        );

        return result.rows;
    }).catch(() => {
```

```
        return false;
    });
}
```

- **Nested Aggregation with Group By** (Line 272 in appService.js):

```
async function findMediaGenreProducer(genreName, producerID) {
    return await withOracleDB(async (connection) => {
        const result = await connection.execute(
            `SELECT mediaTitle
                FROM Media m1
                WHERE m1.producerID = :producerID AND m1.mediaTitle IN
                (SELECT m2.mediaTitle
                 FROM Media m2,
                 WHERE m2.genreName= :genreName)
                 GROUP BY mediaTitle`,
            [genreName, producerID],
            { autoCommit: true }
        );

        return result.rows;
    }).catch(() => {
        return false;
    });
}
```

- **Division** (Line 292 in appService.js):

```
async function findDirectorGenre() {
    return await withOracleDB(async (connection) => {
```

```
        const result = await connection.execute(
            `SELECT D.directorID
             FROM Director D1
             WHERE NOT EXISTS (
                 SELECT G.genreName
                 FROM Genre1 G
                 WHERE NOT EXISTS (
                     SELECT *
                     FROM Media M
                     JOIN Directs D2 ON M.mediaID = D2.mediaID
                     WHERE D2.directorID = D1.directorID AND M.genreName
= G.genreName
                 )
             )`,
        );


        return result.rows;
    }).catch(() => {
        return false;
    });
}
```