Abby Carr, Natalie Duerr, and Kia Zafar

CS3200 - Database Design

Section 1

Group: CarrAbbDuerrNatZafarKia

PROJECT FINAL REPORT

Cinefile Web Application

About Cinefile

Cinefile is an application to locally store films that a user has watched. The project was developed to meet requirements for the final project of CS3200 - Database Design. Running on a MySQL database and a React front-end, Cinefile allows users to learn more about the films in the database, give ratings for films they have watched, and analyze data about their viewing and rating habits.

Team Contributions

Every member of the team worked equally throughout the project with specialized focuses. Abby focused on the MySQL back-end, Natalie worked on developing the front-end, while Kia spearheaded the collection and alteration of data. Throughout the project each team member remained dedicated and worked hard to contribute a near-uniform amount of work.

README Documentation

The README directions for Cinefile can be found on the GitHub repository.

Data Domain

Cinefile focuses on films, their directors, festivals they debuted at, the awards they won, and tracking watched films. The project provides the ability to read all films, information pertaining to a film, and the user's watchlist from the database. It provides the ability to create new films and new instances of a watched film. The project also has the ability to update the information in

or delete an instance of a watched film. The user can add a Festival to a film as well. If the user wishes, they can delete their account, deleting all information pertaining to that user.

Entities, Relationships and Constraints.

All entities in the Cinefile project are identified by an integer ID. The project centers around a film entity that contains the film name, release date, whether it passes the bechdel test, the runtime, the MPAA rating (which can be NULL), and the path to the film's photo. A film is one and only one type of genre and a genre can cover zero to many films. A genre contains its name. A film is directed by one and only one director but a director can direct zero to many films. A director contains their name, birthday, gender, and their race. A film festival entity can debut zero to many films and a film can be debuted at zero to many film festivals. A film festival has a name, location, and year started. A film can win zero to many awards and a type of award can be won by zero to many films. An award includes the award name and the year the award was won. A user can watch zero to many films and a film can be watched by zero to many users. A date and a rating of 0 to 5 (whole numbers only) or NULL is collected when a user watches a film.

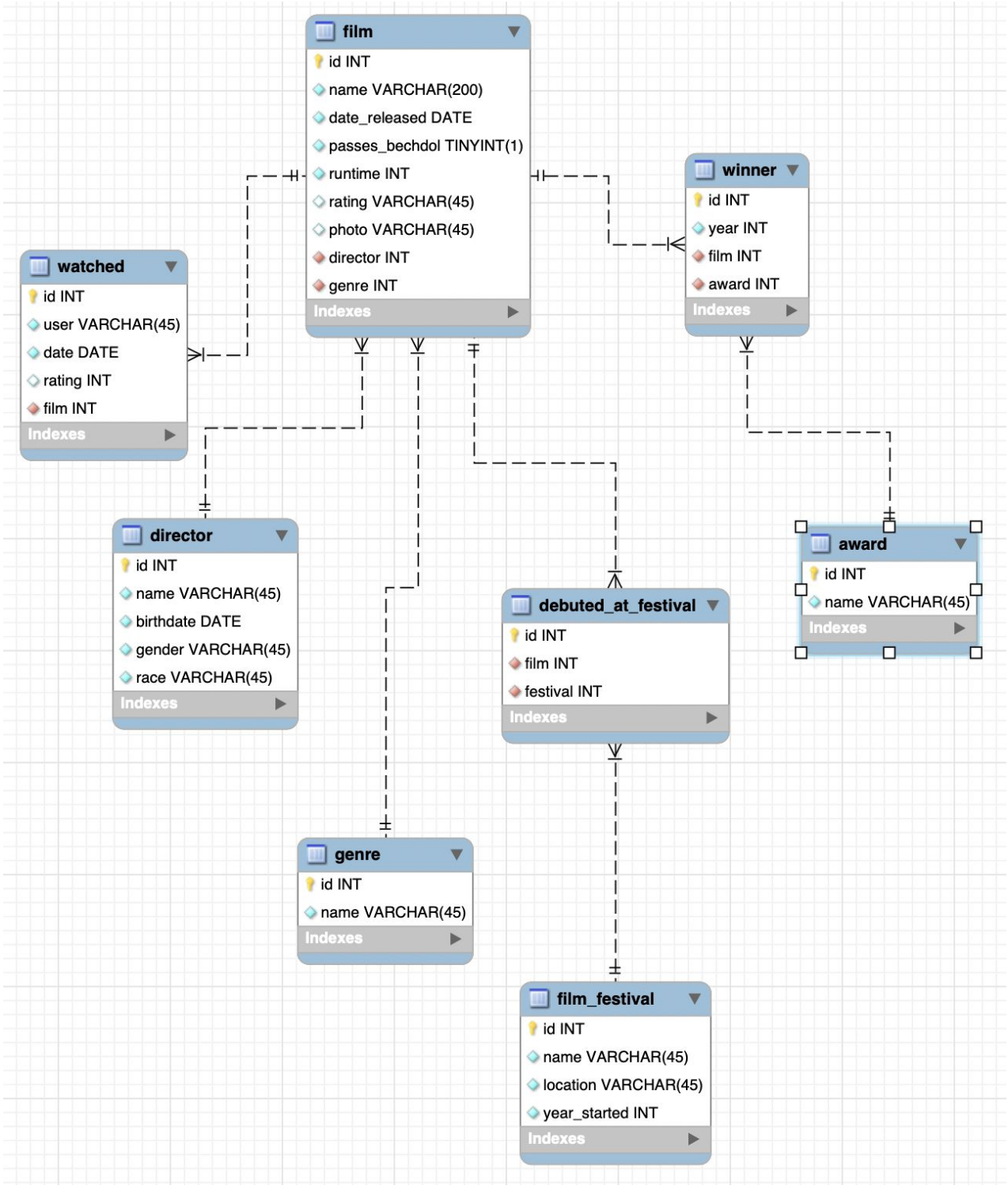
SQL storage

The database is written in SQL and utilized the MySQLWorkbench. The database was built using several SQL files to contain the schema and the other procedural and functional items. After the initial creation, MySQLWorkbench was used to create a dump of the database and all related items to be used by the API and front-end.

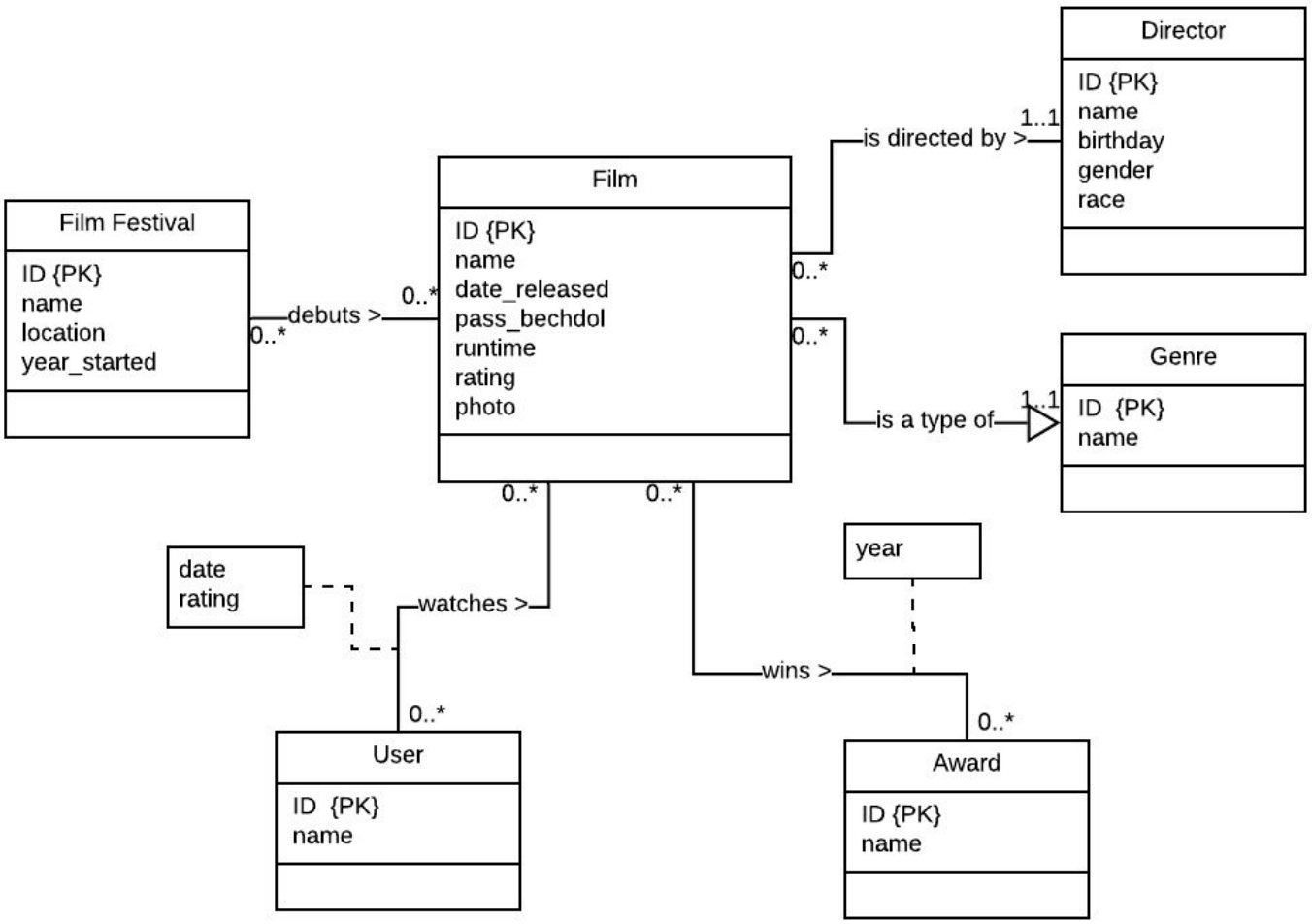
Technical Specifications

This project was built with MySQL, React, Node.js, and Material UI. The front-end code was written in Visual Studio Code while code for the database was written in MySQLWorkbench. The team used GitHub to manage files and Google Meet for online discussions and check-ins.

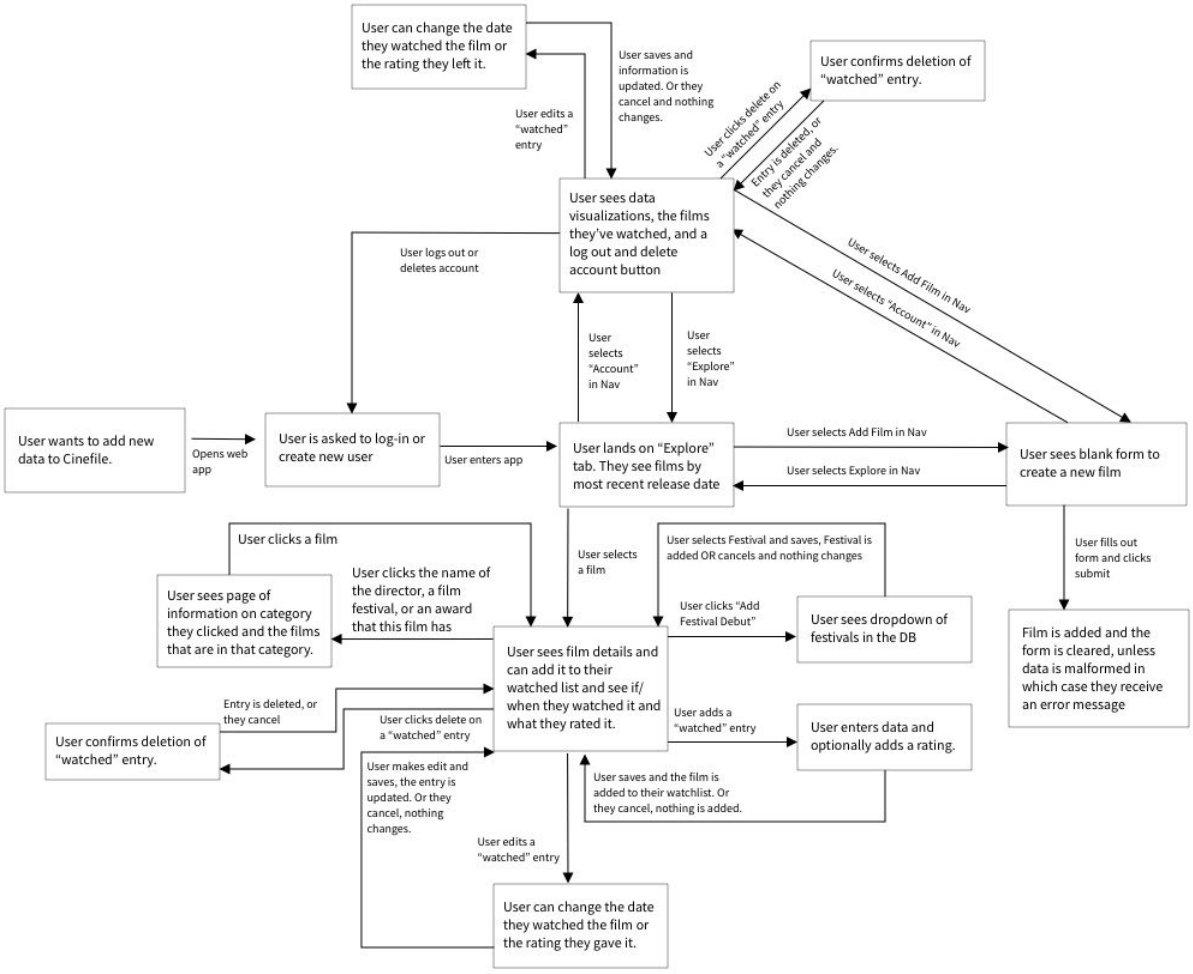
EER Model


film
id INT
name VARCHAR(200)
date_released DATE
passes_bechdol TINYINT(1)
runtime INT
rating VARCHAR(45)
photo VARCHAR(45)
director INT
genre INT
Indexes
winner
id INT
year INT
film INT
award INT
Indexes
watched
id INT
user VARCHAR(45)
date DATE
rating INT
film INT
Indexes
director
id INT
name VARCHAR(45)
birthdate DATE
gender VARCHAR(45)
race VARCHAR(45)
Indexes
award
id INT
name VARCHAR(45)
Indexes
debuted_at_festival
id INT
film INT
festival INT
Indexes
genre
id INT
name VARCHAR(45)
Indexes
film_festival
id INT
name VARCHAR(45)
location VARCHAR(45)
year_started INT
Indexes

UML Model


Director
ID {PK}
name
birthday
gender
race
1..1
is directed by >
Film
ID {PK}
name
date_released
pass_bechdol
runtime
rating
photo
0..*
0..*
Film Festival
ID {PK}
name
location
year_started
debuts >
0..*
0..*
Genre
ID {PK}
name
1..1
is a type of
0..*
0..*
date
rating
watches >
year
wins >
0..*
0..*
User
ID {PK}
name
Award
ID {PK}
name

User Flow



*** The Nav Bar is always visible, meaning the user can navigate to Explore, Add Film, or Account at any time**

The web application relies on the user clicking on buttons and entering in data via the keyboard. Buttons like “SUBMIT”, “+ ADD FESTIVAL”, “EDIT”, “SAVE”, and “DELETE” make it clear that the user can interact with them. Additionally, text links like the director’s name, Festivals, and Awards have a different style to indicate the user can click them.

Lessons Learned

Throughout the project, numerous tools were used to keep the team on track and share materials, which led to many opportunities for personal and knowledge growth. In Cinefile, the user isn't exposed to the SQL file directly and uses inputs to transform data. There were many transformations of data to allow SQL to accept the inputs from the front-end, built using React and node.js. This technology was not something that was taught in class and it was a valuable experience to learn how the programs could interact. As progress was made on the project, team members shared files and data on GitHub which allowed members to learn communication of data often used in professional settings. There was much learned both about the structure of GitHub as well as how the team could improve productivity through individuals working on materials at the same time while remote.

As the team split up during the coronavirus evacuation, there were many road bumps including different time zones, bad Wi-Fi, and the lack of in person communication. Working together was limited to video calls scheduled online. It was difficult to develop a consistent schedule that worked for each individual and gave the team enough time to complete the project. Working as a team allowed the members to build time awareness skills both for when meeting times would be most efficient and when each individual would work on their own tasks. Members learned a lot about when they could meet their deadlines and their own strategies for coping while working during a pandemic. In addition, working through the project helped members find their own strengths and built a foundation where each member was entrusted with responsibilities that best fit those strengths.

When working with the data domain for this project, some issues concerning the data were found, though most were minor. As the data domain stemmed from Oscar Best Picture winners, the largest issue found was the occasional presence of co-directors, which was not included in the project. The team's solution was that one director had to be chosen to be displayed for a film. Another issue stemmed from time; since some of the award winners had been produced before the MPAA ratings existed. The database had to allow for null ratings for a given film to account for this. This was an issue that was easily acceptable for the database but did provide less constraint on the properties of a single film.

As the project was created, an image of the user interface was formed but there were other directions available. For example, a big change could have been the format of the user interaction. To accommodate for a team size of three, the user interacts with a web application. This setup garnered a large amount of work, but allowed for more freedom in design; to alternate the design the project could have been formatted as a smaller and more limited command line project. On the other hand, to change some of the project's structure the main explore tab could have been sectioned, either by director, award, or film festival. The current design applies a cyclical and linked view of the database but could have had many different read and exploring structures.

The lessons of this project were impactful to the team. The project does not include any code that does not work. The project has many capabilities to be explored that are discussed later in the

report. However, this would have been additional functionality and the team prioritized functional and clean code.

Future Work

In the future, there is functionality that Cinefile could gain. These features include some options more related to the database such as checking if a film exists before adding it, being able to use filters to affect movies being viewed on the Explore page, and the ability for the user to add new directors, genres, festivals, and awards. There are also some things that could be added in order to improve the user experience, such as adding the need for a safely hashed password for security, as well as making the site more responsive to platforms besides desktop and laptop. The database could be expanded to initially include more films and directors so that the user does not need to create them.