Brendan Gressel

Stephen Johnson

Tyler Yelverton

Stage 4: Databases Project Report

For this project, we created a database that sorts and compares book to movie adaptations and uses online resources to initially establish scores that the user will be able to view simultaneously. Our database contains books with their title, rating, and author, while our films have their title, rating, and year of production. All book ratings and places of publication were originally taken from Goodreads, while the films ratings and production companies were taken from IMDB. We chose this topic because we love books and movies and have yet to come across a website that does this comparison. We thought it would be interesting to see the scores compared from general audiences and finally answer the question, was the book better than the movie? This initially works as a one-stop website for people looking for easy comparisons. This website will also allow users to share their opinions of the adaptations. The key components for our website are laid out in three columns. The middle column holds the title which the user searched up, to the left is the book rating and publisher, and the right is the movie rating with its production company. Above these columns is a search bar where the user can make queries, and below is a submission form for the user to submit their own ratings.

**\*UPDATE AFTER FINISHED\* Just the changes to the attributes**

We designed our database using three different entities: TV Show/Movie, Book, and Production Company. The TV Show/Movie entity has a Title key, as well as a Rating and Year attribute. It has an “adapted from” relationship with the Book entity and a “produced by” relationship with the Production Company entity. The Book entity has a Title key, as well as a Rating and Author attribute. It has an “adapted to” relationship with the TV Show/Movie entity and a “published by” relationship with the Production Company entity. The Production Company entity has a Name key as well as a Location attribute. It has a “published” relationship with the Book entity and a “produced” relationship with the TV Show/Movie entity. For the relationships, the TV Show/Movie “adapted from” Book relationship is many to one because we assumed all adaptations must come from one Book, however a Book can be adapted many times. The TV Show/Movie “produced by” Production Company relationship is many to one because we assumed each TV Show/Movie only has one Production Company, but a Production Company can produce many TV Shows/Movies. The Book “published by” Production Company relationship is the same, as we assumed a Book only has one publisher, but a Production Company can publish many Books.

**\*UPDATE ER DIAGRAM AND SCHEMA\***

ER Diagram:



\*We assumed there is only one publisher per book.

\*We assumed there is only one studio per TV Show/Movie.

\*The arrows should be pointing into the production company from the Book and TV Show/Movie Entities.

Relational Schema:

TV Show/Movie = (T\_Title, Year, T\_Rating, B\_Title, P\_Name)

Book = (B\_Title, B\_Rating, Author, P\_Name)

Production Company = (P\_Name, Location)

**\*FINISH DATABASE DETAILS AFTER FINISHED\* What are the tables you include in the database? How do you design your database to accommodate functional dependencies? Are your tables in BCNF or 3NF? How about other constraints?**

Our basic features involve searching, inserting, updating, and removing current records in the database. To insert records, we will require someone to have a movie and a book in order to validate the entry. They will need to provide the production companies and ratings, which they can get from IMDB and Goodreads or be personal, when entering their adaptations. In order to search the database, the user will need to know the title of the book and movie pair because those are the keys for our entities and are required for registration. Some examples of this would be: The Shining (Film, 8.4 on IMDB, Warner Bros.) vs The Shining (Book, 8.4 on Goodreads, New English Library). To update records, we will allow users to add their own ratings to the books or movies when searched. These will then be separated into different ratings based on whether the user has read the book or not. Since titles and production companies do not change, they cannot update those. To delete an entry: if it were a movie, it would not delete the book, however if the book were deleted, all adaptations would also be deleted. Our advanced function is comparing ratings among book readers and moviegoers based on whether they’ve read the book or not. We will allow the user to sort the entries by the difference between the movie rating (from those who have read the book) and book rating, thus showing if the adaptation was a good or bad one and if it exceeded in quality over the book. **\*ADD HOW WE IMPLEMENTED\***

**\*ADD IMPLEMENTATION DETAILS\* What languages and platform you’ve chosen to do the implementation? How do you implement the front-end Web interface and the application logic? How does the front-end Web interface interact with the backend database? Students are encouraged to upload their code to some online repositories, such as Dropbox, Github, Bitbucket, and provide the link in the report.**

We learned a lot from this project, especially how to organize databases. We learned how to code in SQL and PHP, as well as integrating the two together so that they work in unison. The structuring of the tables and how they are organized was a large part of our learning, as well as displaying these tables on the website. Adding in our basic features and our advanced features was another hurdle, as we had to use HTML to get the data from the user and then use PHP to parse it through the database. A challenge that occurred was when we were trying to implement our advance feature that separates user ratings based on whether they have seen the movie or not. **\*ADD HOW WE SOLVED THIS\*** We will use what we learned in this project to make more complicated systems in the future. Now that we understand how they are implemented and organized, we feel very comfortable making systems with more than three entities and even more relationships and attributes. We feel comfortable with the languages used, and granted they are used in whatever professional workplace we end up at, the experience will be a nice reminder of how to use them and we will easily be able to prompt ourselves on how to work with them.