The Amazing InfoDB

Final Report

Tsz Hei Leung
Sheel Parekh
Tim Klabjan
Brendan Wilson

University of Illinois
CS 411
Group 8: Little Bobby Tables
Dec 2016

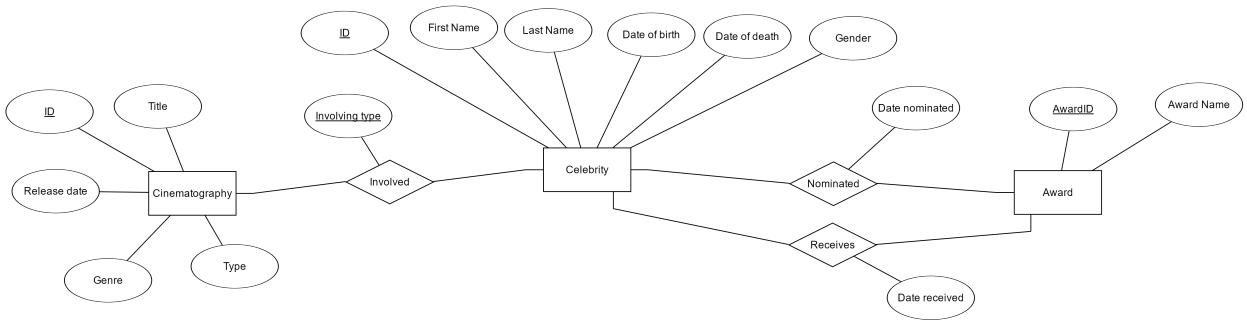
Table of Contents

Initial Development

Project Idea

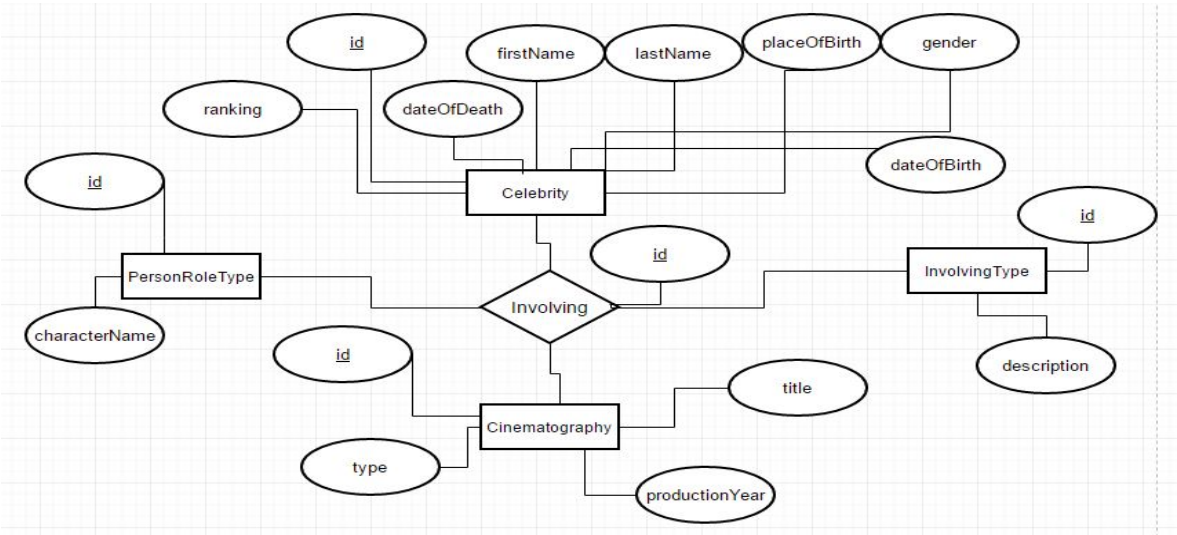
InfoDB is an online webapp that aims to show detailed information about celebrities and movies to the users. InfoDB's defining characteristic is its focus on how actors are related and connect to each other; it includes a Network Explorer and Degrees of Separation Finder that allows users to explore the data in different dimensions. Potential users that can be benefited from this app include movie lovers, actors, producers, and fun fact enthusiasts.

ER Design and Schema Design



Initial ER Design

Our initial ER design includes Award, but was later removed due to the fact that data on awards are not present in the datasets provided by IMDb. The following is our modified schema in response to the issue:



Current ER Design

Since we have implemented indexing for every table, each table has an Id field that functions as the primary key, whereas some attributes of the Involving table are foreign keys (can be identified in *italics* below).

```
Celebrity (
      Id,
      PlaceOfBirth,
      DateOfBirth,
      DateOfDeath,
      FirstName,
      LastName,
      Gender,
      Ranking
)
Cinematography (
      Id,
      Title,
      ProductionYear,
      Type
)
Involving (
      Id,
      CinematographyId,
      CelebrityId,
      InvolvingTypeId,
      CharacterNameId,
      RoleDescription
)
InvolvingType (
      Id,
      Description
)
PersonRoleType (
      Id,
      CharacterName
)
```

Main Development

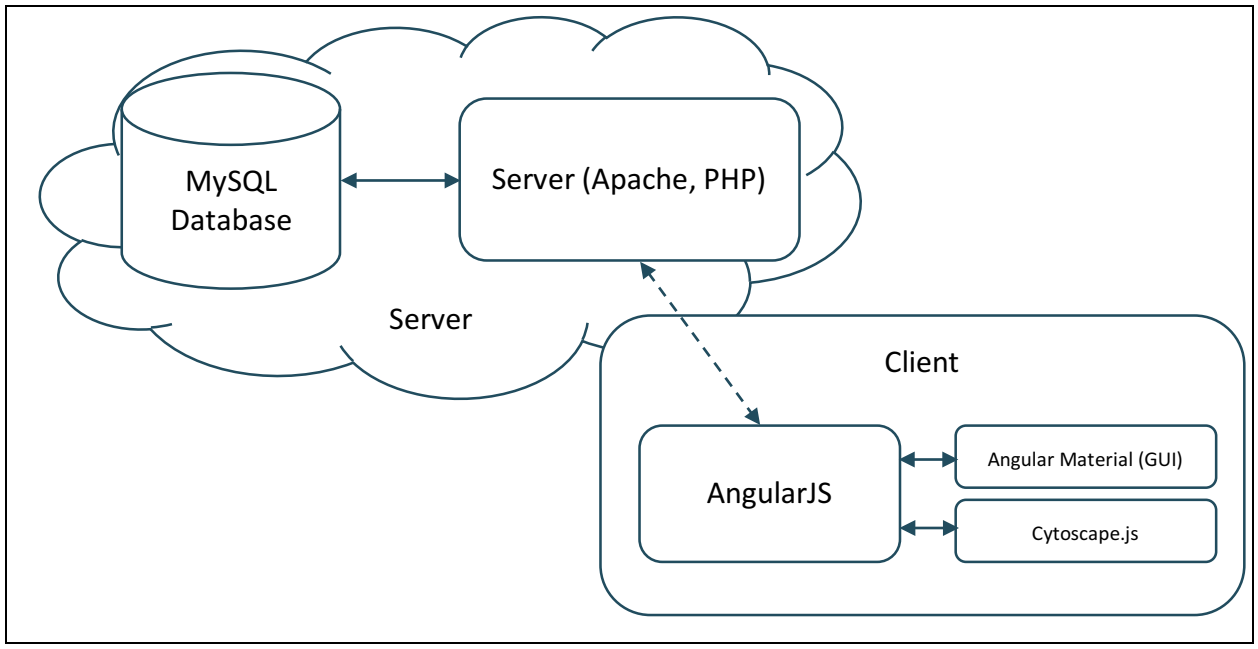
Importing Data

The dataset provided by IMDb is initially in a format that is not nicely compatible with our MySQL database. We ended up using a tool called “IMDbPY” to help transform and import the actual data into the correct format and create the corresponding tables. It also helped us in generating valid IDs for entries that we were then able to use to uniquely identify different entities such as celebrities and cinematography.

The Data

The tables and their attributes are listed above. Most of the tables contain a couple hundred thousand entries with the `Involving` table being the largest. As was first noticed when trying to import the raw IMDb data into the database, the final imported data is not the cleanest dataset out there. The two biggest issues present throughout the dataset are missing data and “junk” entries. For the missing data, some are structurally missing, for example a date of death for a celebrity who’s still living is not applicable. Other missing attributes however, such as place of birth, are either unknown or not provided by IMDb. The other component of the dataset that adds to its non-cleanliness is the presence of “junk” entries. For celebrities, “Tom Cruise” is correctly present, but “Tommy Cruise” is there as well. The inclusion of such entries unnecessarily increases the size of each dataset.

Development Environment



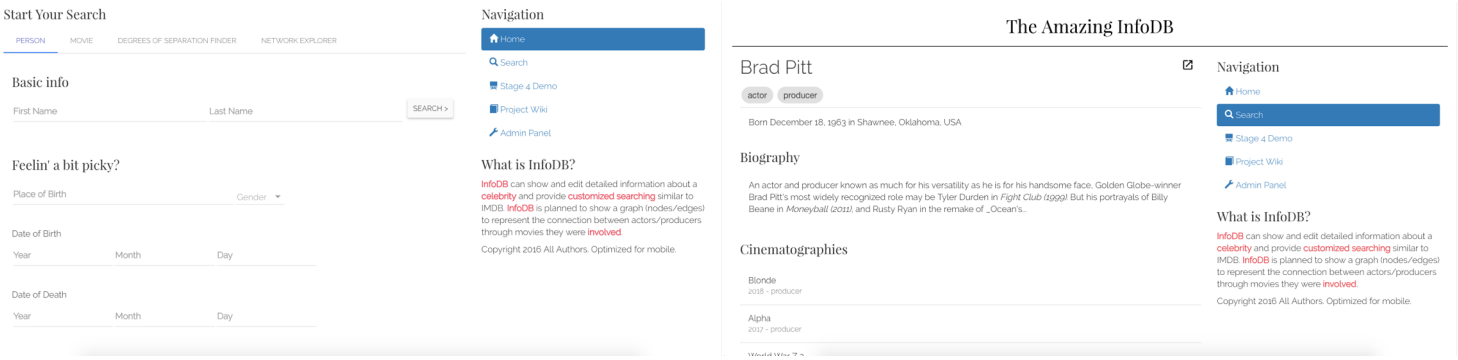
We chose to work with the LAMP stack because it is easiest for everyone to adapt to. PHP might not be the best server-side language, but it gets the job done.

For the client side, we chose to work with JavaScript along with AngularJS and Angular Material for the Material styling. We also use Bootstrap for easy mobile integration. JavaScript is used since using another language requires additional steps for transpiling the source code, which is unnecessary for our project. AngularJS is useful for displaying data and helps with creating declarative HTML documents. Its data binding capability allows us to easily display different information without directly modifying the DOM. It also handles a great part of the server-client communication. Angular Material plays nicely with AngularJS, and it provides a lot of useful pre-made components that sped up our development by eliminating having to spend time on the cosmetic aspect. Cytoscape.js is used to help generating graphs quick and efficiently.

Features

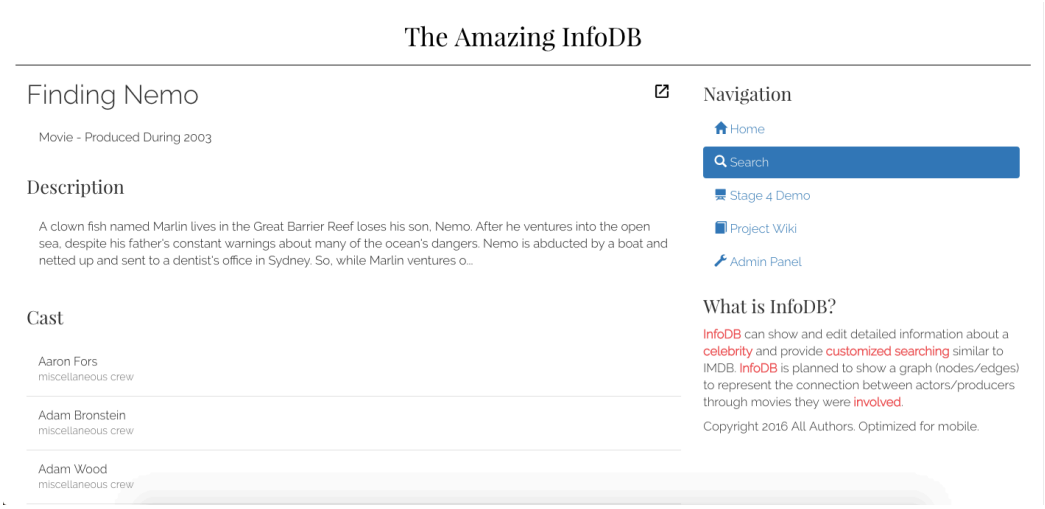
Basic Functionalities

The two basic functionalities of InfoDB are displaying information about an actor or movie that the user is interested. For an application that desires to inform users about actors



and movies, these two functionalities are the core features and are what IMDb is built upon. The first feature simply involves the user specifying a celebrity and then the application displays basic information about the celebrity. The user can also specify things such as gender and place of birth when performing the search.

The other basic functionality is similar to the actor search, just with movies. Obtaining the necessary information for this feature requires joining tables in the back-end SQL query.



Search View

The Search View takes in a wide range of parameters from names to birth places. These parameters are passed to the server side PHP code which will then dynamically build a SQL query in the following format:

```
SELECT c.firstname, c.lastname, i.description
FROM Celebrity c LEFT JOIN CelebrityInfo i ON c.id = i.celebrityId
WHERE *conditions*
ORDER BY LENGTH(i.description) DESC, c.firstname ASC, c.lastname ASC
LIMIT ?,?
```

The conditions can be any of the following:

```
c.firstname LIKE ?
c.lastname LIKE ?
c.placeOfBirth LIKE ?
c.gender LIKE ?
YEAR(c.dateOfBirth) = ?
MONTH(c.dateOfBirth) = ?
DAY(c.dateOfBirth) = ?
YEAR(c.dateOfDeath) = ?
MONTH(c.dateOfDeath) = ?
DAY(c.dateOfDeath) = ?
```

The results of the query are ordered by the length of the person's biography, which is often a good indicator of the popularity.

The implementation can be found in `/search/person/index.php` or `/search/movie/index.php`.

Detail View

The Detail View fetches all available information of an entity based on an ID. The PHP code will dynamically generate a new SQL query according to the input in the following format:

```
SELECT
    c.id, c.firstname, c.lastname,
    IFNULL(i.description, "No biography."),
    c.gender, c.placeOfBirth, c.dateOfBirth, c.dateOfDeath
FROM Celebrity c LEFT JOIN CelebrityInfo i ON c.id = i.celebrityId
WHERE c.id = ?
```

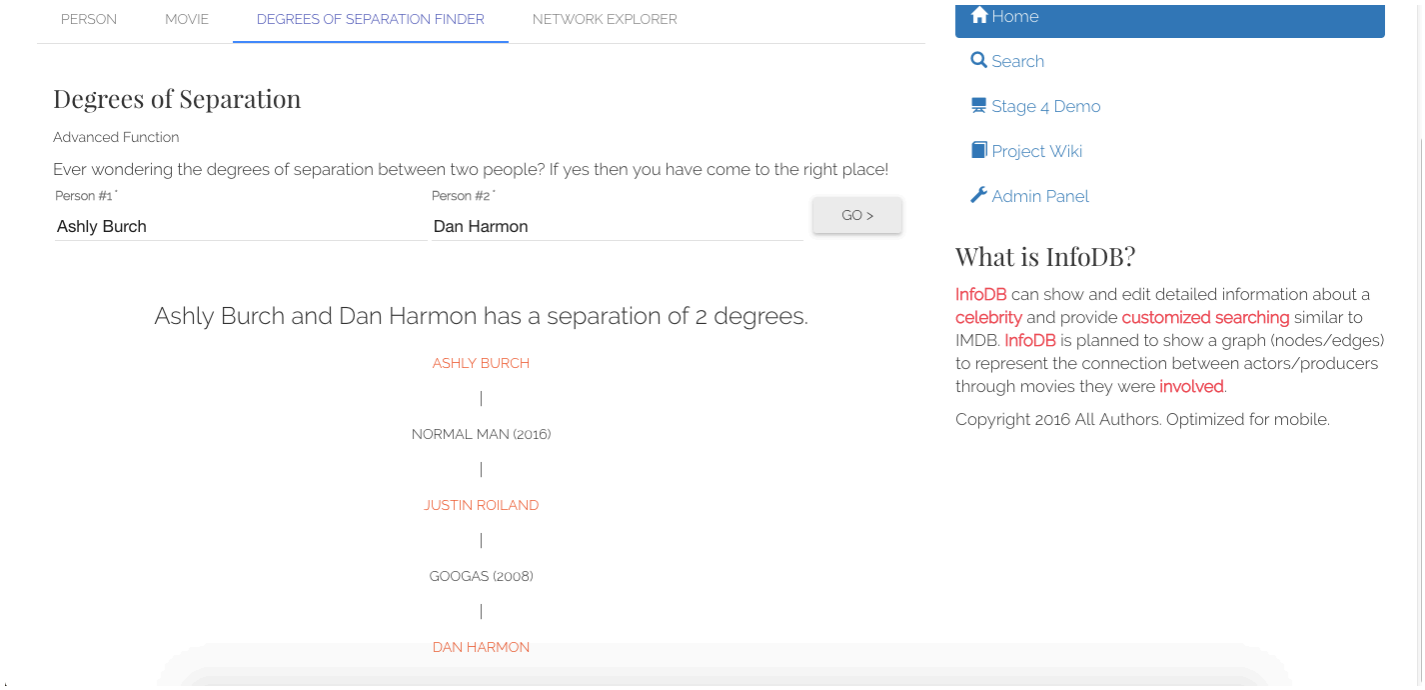
Advanced Functionalities

Degrees of Separation Calculator

The advanced functionalities center around displaying the connectedness of actors. This is a unique feature of InfoDB that extends beyond the usual simple display of text containing information about a specific movie or celebrity. The functionalities provide information on how actors are linked together, either by working on a movie set together directly or through mutual coworkers.

The first advanced functionality is the Degrees of Separation calculator. It's a utility that allows the user to find the degrees of separation between any two celebrities. The result is a simple interface depicting a chain of celebrities that shows how they are related and what

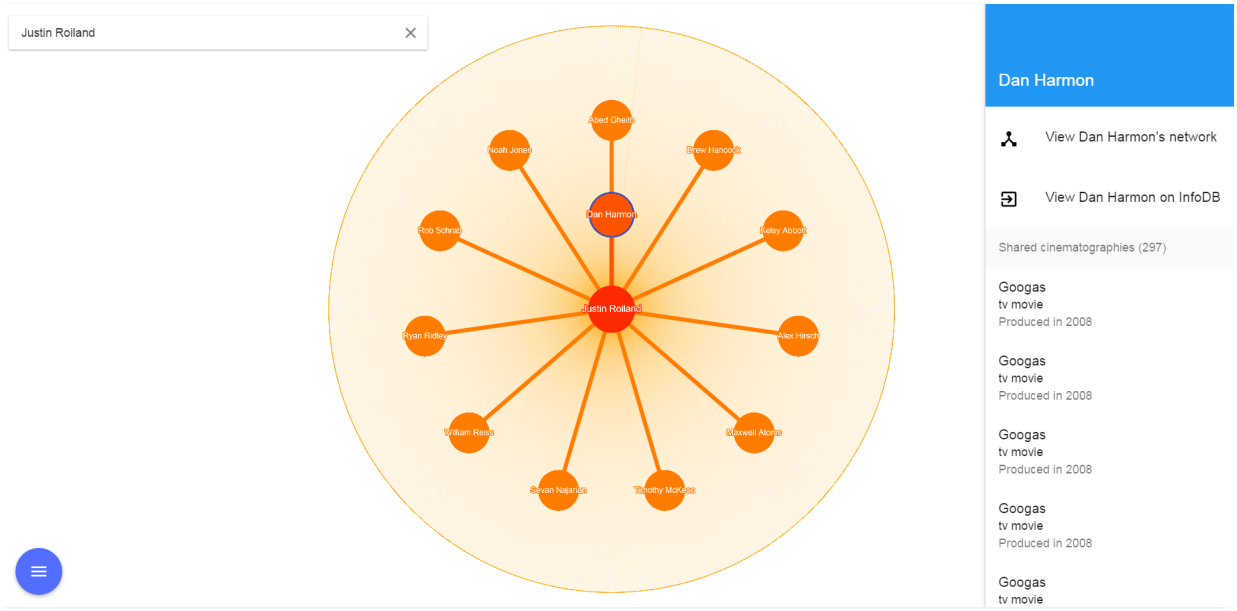
cinematography (movie, film, etc.) each pair of celebrities was involved in. The implementation of this feature was mainly done in SQL using a procedure (SQL equivalent of a function) and involves potentially many increasingly larger table joins. The core algorithm is finding all the actors the start actor has worked with, and then finding all the actors those actors have worked with until the second target actor is found. With each increase in degree of separation, a temporary table needs to store pairs of actors and how they are related. These temporary



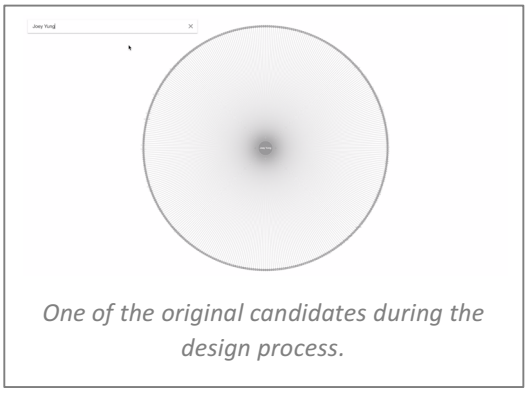
tables can explode in size quite rapidly, and the algorithm can take undesirable time to execute. We theorize more efficient solutions in the future work section, but they do not involve concepts presented in this class such as advanced SQL queries, which we used heavily.

Designing an interface to display the necessary data is not difficult since the amount of data returned is very limited. As a result, we have chosen to display the data through a simple text-based interface that is easy to understand for an average user. Names of celebrities have different colors than the names of movies to emphasis the differences between these two types of entities.

Network Explorer



A sample graph with 12 main nodes. Note that the outer ring is actually a ring of tiny nodes.

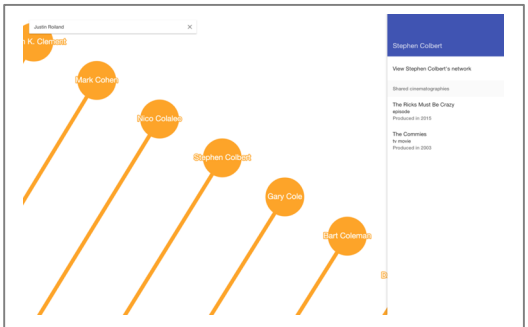


One of the original candidates during the design process.

The other advanced functionality is the *Network Explorer*. It's a utility that presents a graphical visualization of an actor's neighborhood in the connectedness graph, i.e. those actors with whom the focal actor has a degree of separation of one. It presents the focal actor in the center of a radial graph, with the others arranged around them. The others are arranged into tiers of importance based on how many films in which they have appeared with the focal actor.

The implementation of the *Network Explorer* involves a single SQL query to find the actor's neighborhood based on our `Involving` relation, some PHP to rearrange and format the query into JSON, and a client-side display using the Cytoscape.js library to render the graph. The server-side code is generally quick, however rendering larger graphs for more prolific actors can take some time.

One of the biggest challenge in displaying the data is that there is simply too much information to display at the same time. Some celebrities have over 20,000 connected nodes, which if we simply uniformly place the nodes equally in size in the stage, it would be extremely hard to navigate and users will not be able to extract any useful information which renders the tool useless. A solution that our team



Infinite zooming allows us to correlate size with relative importance and connectivity.

came up with is to scale the nodes by the weight of the edges. Actors and directors that are more connected will show up larger to catch the attention of the users. If the user is interested in inspect other loosely connected actors, they can zoom in to the outer levels to take a look at the individual nodes. The number of nodes can easily be configured through a single variable in the implementation. The sizes and colors of each level are computed via a linear map.

Future Work

The one issue with our current application that we would like to improve upon in the future is the speed of the degrees of separation calculator. We feel like we have made the most efficient algorithm we can for an SQL joins based algorithm, but it still can take significant time to execute due to slow connections and large tables being joined. We know that a much faster algorithm exists.

This algorithm would involve first building a network graph where each node is an actor and two nodes have an edge between them if the two actors worked in a movie together. This graph would be large and building/storing it might take time, but this step only has to be done once. Upon a user's degrees of separation calculator request, the two nodes corresponding to the inputted actors would be found. After this, calculating the degrees of separation requires simply running a shortest path algorithm between the two nodes such as Dijkstra's algorithm. The runtime of Dijkstra's would certainly be much faster than the current SQL-based version. However, this is a database systems course and not a graph theory course, which is why the SQL algorithm was implemented despite being non-optimal.

Logistics

Work Division Among Team Members

Sheel Parekh

- ER Diagram/Relational Model
- IMDB Degrees of Separation (Advanced function) API

Tim Klabjan

- Proposal and final report, video demo
- Helped with functionalities

Tsz Hei Leung

- Front end design and development
- Helped on some backend PHP and SQL coding.

Brendan Wilson

- Data import and mangling
- Network Explorer backend
- Idea spitballing

Lessons Learned about Teamwork

We were fortunate enough to find a great team that did not have any issues with each other. Any problems encountered throughout this project were all technical challenges and not problems with teamwork. All group members showed up to group meetings and put forth their fair share of effort to ensure the project's deadlines were met. The lesson learned here is that

group projects are way more fun and efficient when the project itself is the only focus; no extraneous distractions arising from issues with teamwork were present.