Project Outline

CIS 5500

Team Members

Brendan Brett - bbt@seas.upenn.edu - @brendanbrett
Edoardo Palazzi - palazzi@seas.upenn.edu - @EdoardoPalazzi
Eitan Jacob - eitanj@seas.upenn.edu - @EitanJacob

Git Repository

https://github.com/brendanbrett/cis5500-project

Outline: Give a detailed functionality description and schema design, and initial set up.

Motivation for Idea

Motivation for the idea/description of the problem the application solves

An *EGOT* refers to someone who has won each of the four major American performing art awards: '**E**mmy, **G**rammy, **O**scar, **T**ony'. The motivation for the application idea was to create an authoritative source database that identifies those legendary artists, and to quickly allow users to identify the EGOTs and which awards they have won. Through our review of existing sources online, we have found static articles that list out EGOT winners, but we have not found any websites which aggregate this information in a systematic way and display all current winners.

Features

Definite Features

List of features you will definitely implement in the application

1. EGOT award winners list: identify all Artists who have are designated as EGOTs
2. EGOT details: drill in on the EGOT-winning artist and see which awards they have won
3. EGOT Award Winner Analysis & Trivia: answer 5 trivia questions such 'who among the top 100 nominees was nominated the most times without winning before finally receiving an award?'
4. Award list page: for each of the major awards, provide a comprehensive list of all award years, award categories, nominees, and winners.
5. Sorting and Filtering on award list page: Ability for user to filter the data for an award list page i.e., filter on award year, nominee, winners, award category.

Stretch Features

List of features you might implement in the application, given enough time

1. Scrollable timeline: create a scrollable visual timeline for each major performing art award. This would allow users to scroll through the 'history of movies', 'history of music', 'history of broadway, and 'history of TV'.
2. Knowledge Challenge - Similar to sporcle.com, provide quizzes with trivia questions to test the user's knowledge. These quizzes can come in various forms (e.g. multiple choice, matching). See end of this outline where 6 such sample trivia questions are listed.

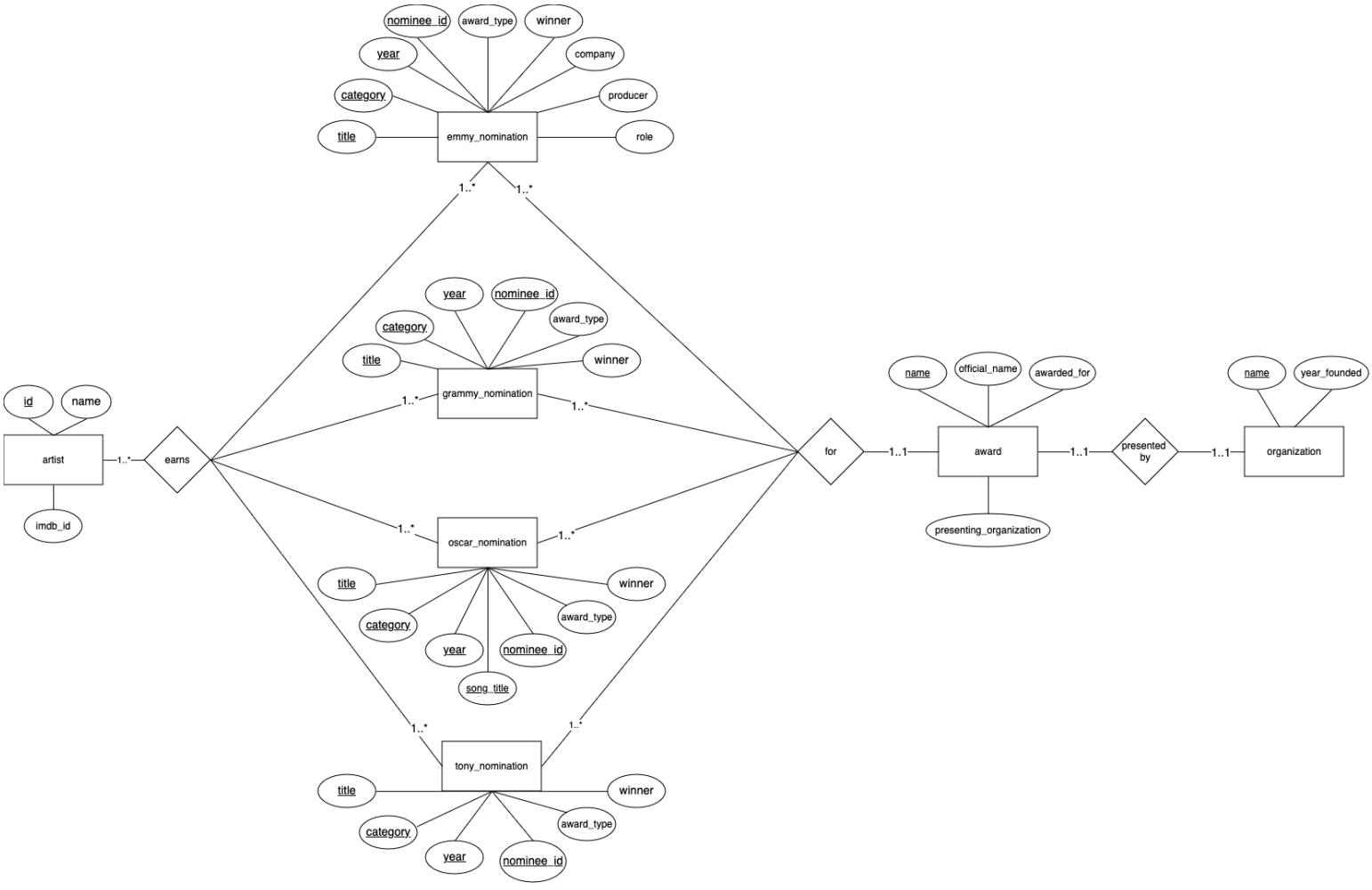
Pages

List of pages the application will have and a 1-2 sentence description of each page. We expect that the functionality of each page will be meaningfully different from the functionality of the other pages. (At least 3 pages)

1. Overview page: List page of EGOT winners. This page will be the index page of the site and will provide a list of all EGOT award winners, sorted chronologically. Each of the individual artists will be a link to the EGOT detail page.
2. EGOT Detail page: This page will provide details of the EGOT-designated artists award wins, listing out which year they were designated an EGOT, and a list of the major awards they have won.
3. EGOT Award Winner Analysis & Trivia: This page will allow users to view the Analysis & Trivia answers based on the EGOT award winner data.
4. Award List Page: This page will provide the entire history of awards for a particular major award. For example, if a user views the Oscars, this would be a list of all years, all award categories, all nominees, and all winners.
5. (Stretch Page): a scrollable visual timeline for each major performing art award. This would allow users to scroll through the ‘history of movies’, ‘history of music’, ‘history of broadway, and ‘history of TV’.

Relational Schema

Relational schema as an ER diagram



SQL DDL for creating the database

SQL DDL for creating the database

CREATE DATABASE final_project_egot;

```
CREATE TABLE organization (
        name VARCHAR(100) PRIMARY KEY,
        year_founded INT
);

CREATE TABLE award (
        name VARCHAR(100),
        official_name VARCHAR(100),
        awarded_for VARCHAR(100),
        presenting_organization VARCHAR(25),
        PRIMARY KEY (name)
        FOREIGN KEY (name) REFERENCES organization(name)
);

CREATE TABLE artist (
        id INT PRIMARY KEY,
        name VARCHAR(100),
        imdb_id INT
);

CREATE TABLE emmy_nomination (
        year INT,
        category VARCHAR(100),
        title VARCHAR(50),
        nominee_id INT,
        role VARCHAR(50),
        company VARCHAR(50),
        producer VARCHAR(50),
        winner BOOLEAN,
        award_type VARCHAR(25),
        PRIMARY KEY (category, nominee_id, title, year)
        FOREIGN KEY (award_type) REFERENCES award(name)
        FOREIGN KEY (nominee_id) REFERENCES artist(id)

);

CREATE TABLE grammy_nomination (
        year INT,
        category VARCHAR(100),
        title VARCHAR(50),
        nominee_id INT,
        winner BOOLEAN,
        award_type VARCHAR(25),
        PRIMARY KEY (category, nominee_id, title, year)
        FOREIGN KEY (award_type) REFERENCES award(name)
        FOREIGN KEY (nominee_id) REFERENCES artist(id)
);

CREATE TABLE oscar_nomination (
        year INT,
        category VARCHAR(100),
```

```
        title VARCHAR(50),
        nominee_id INT,
        winner BOOLEAN,
        song_title VARCHAR(50)
        award_type VARCHAR(25),
        PRIMARY KEY (category, nominee_id, title, year, song_title)
        FOREIGN KEY (award_type) REFERENCES award(name)
        FOREIGN KEY (nominee_id) REFERENCES artist(id)
);

CREATE TABLE tony_nomination (
        year INT,
        category VARCHAR(100),
        title VARCHAR(50),
        nominee_id INT,
        winner BOOLEAN,
        award_type VARCHAR(25),
        PRIMARY KEY (category, nominee_id, title, year)
        FOREIGN KEY (award_type) REFERENCES award(name)
        FOREIGN KEY (nominee_id) REFERENCES artist(id)
);
```

Pre-processing

Explanation of how you will clean and pre-process the data.

We initially came up with the app idea after finding the Grammy's and Oscar's datasets on Kaggle, however we ran into two issues:

1. After cleaning and processing the data, and performing Exploratory Data Analysis (EDA) using a Jupyter Notebook (via Google Colab), we were generally unhappy with the quality of the datasets. We also found that they were not comprehensive enough.
2. We could not find datasets for *all* awards, only a subset, which would have prevented us from identifying an EGOT award winner as we needed 5 different award datasets (due to Primetime Emmys and Daytime Emmys being technically different awards, eve though they are counted the same for EGOT purposes).

To ensure our app idea could become a reality, we created web scraping and processing tools for *every* award category, scraping the data from the 4 different official award websites (in combination with imdb.com specifically for Daytime Emmy data), which allowed us to create our own new, high quality datasets, based off of the latest information available!

We subsequently performed EDA on our new datasets again using our Jupyter Notebook (specifically, utilizing pandas), and it has been extremely efficient in helping us to identify anomalies in our data so that we can subsequently investigate the cause and enhance our web scraping and processing tools as needed.

List of technologies

List of technologies you will use. You must use some kind of SQL database. We recommend using MySQL specifically because you will use MySQL in HW2, and we will provide guidance for setting up a MySQL database.

1. Python (pandas) & Jupyter Notebook (via Colab) for EDA
2. Python for web scraping and data processing

3. React for front-end development
4. Node.JS for server-side development
5. AWS RDS for hosting and management of our relational database
6. MySQL database
7. Github for version control system

Responsibilities

Description of what each group member will be responsible for.

Brendan Brett

1. Project architecture & design
2. Scraping of primary data sources: Grammys.com, Oscars.org, IBDB.com (Tony's)
3. Data processing for primary data sets: Emmys, Grammys, Oscars, Tony's
4. Front-end UI design
5. API integration
6. React components and templates

Edoardo Palazzi

1. Scraping of Emmys (daytime) data: imdb.com
2. SQL DDL & Relation Schema (ER diagram)
3. EDA on initial datasets + data cleaning/pre-processing on certain final datasets
4. Build front-end UI features
5. Building SQL queries for the various functionalities

Eitan Jacob

1. EDA on datasets - identified dataset anomalies and addressed inconsistencies within the datasets, ensuring all missing EGOT winners are included
2. Query Preparation - prepared a comprehensive list of queries.
3. Scraper Enhancement - improved Tony's scraper to efficiently gather all desired data.
4. Contribute to UI Design
5. Designing algorithms for the sorting/filtering page