Design Document

Project 1, Part 1

Team 01000101

Landon Creel, Taylor Gray, Buster Lee, Nicholas Yannuzzi

CSCE 315-909

Section 1: Purpose of System

Problem Statement

The Department of Film and Media needs an application that features a graphical user interface (GUI) for managing and querying data from the Internet Movie Database (IMDb). The department needs a database management system (DBS) for storing data information from media, cast and crew. The application should allow students to input names of actors, which would be used as queries to generate trivia questions for class assignments.

Relevance

The reason for creating the system is to propagate film media information to the students to aid their reticence of course material through trivial questions. This system needs to exist due to rapid decline of cultural and artistic diaspora in American culture, thus the students in the Department of Film and Media can be the iridescent beacon to the macabre troglodytes with no media awareness.

Target Users

The primary target users for this application will be students taking the introductory film studies course in the Department of Film and Media at Texas A&M University.

Section 2: High-Level Entity Design

People

- Purpose
    - Each object in this entity will contain things such as name, age, and type of person (whether that person is in the cast or the crew). It will be used to identify who worked on or acted in a specific movie.

- Justification
    - There must be a people object in the database because movies contain people. This entity will also make the database structure more general. It will allow us to utilize the DRY (Don't Repeat Yourself) technique.
- System Role
    - The role of the people entity is to abstract the different types of people in a movie.

Movies

- Purpose
    - Each piece of data in this entity will contain pertinent information about the move such as genre, year of release, rating, and studio. Different people who worked on the movie will also be present in this object.
- Justification
    - The introductory film studies class in the College of Liberal Arts would not be complete without films/movies.
- System Role
    - The movie grouping aggregates the people that were a part of a movie.The purpose of having a people entity is to abstract the different types of people in a movie.

Section 3: Low-Level Entity Design

Movie

- Relationship
    - Container class that holds categorical data about the movie as well as a list of cast and crew that worked on the movie. A movie has many people, and each person has 1 or more movies.
- Thought Process
    - There has to be an entity to represent a movie, and this movie also has details such as cast/crew and other information.
- Benefits
    - The benefit of having a movie class is to represent a film and to encapsulate people involved in the film.
- Risk
    - There is zero risk in creating Movie as an entity. It is absolutely necessary in this project.

People

- Relationship
    - People is an abstract class which will be extended to form several child classes: cast, or crew. People belong to 1 or more movies.
- Thought Process
    - Having an abstract class to represent people will be necessary; the class will also allow people to be associated with their movies, simplifying searching.

- Benefits
    - Having an abstract class will greatly simplify searching and allow much of the background implementation to be shared between the various types of people. Without an abstract class, we would have to separately define each type of person, wasting resources since much of their needed functionality is shared.
- Risk
    - The risks of using an abstract class is that it might overcomplicate the project, but no more so than defining each type of person as their own class.

Cast

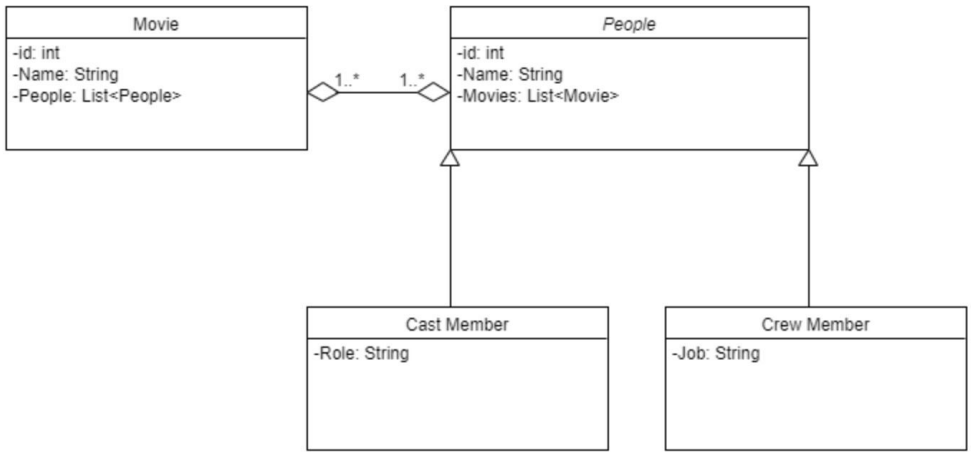
- Relationship
    - Cast is one of the child classes of people that keeps its own separate variables from the other child class, crew.
- Thought Process
    - Having a child class would help to keep the actors and the crew separate because they have different jobs relating to the movie.
- Benefits
    - The benefit of having the child class of cast is that it can have its own data value known as job, since they only act in the movie.
- Risk
    - A risk of having a child class is having to manage the data flow between the parent and the child class.

Crew

- Relationship
    - Crew is one of the child classes of people that keeps its own separate variables from the other child class, cast.
- Thought Process
    - Having the crew child class helps to keep the crew jobs separate from the actor jobs since they are much different.
- Benefits
    - The benefit of having the child class of crew is that it can have its own data value known as job, since they do not act in the movie: they do the outside jobs such as editing.

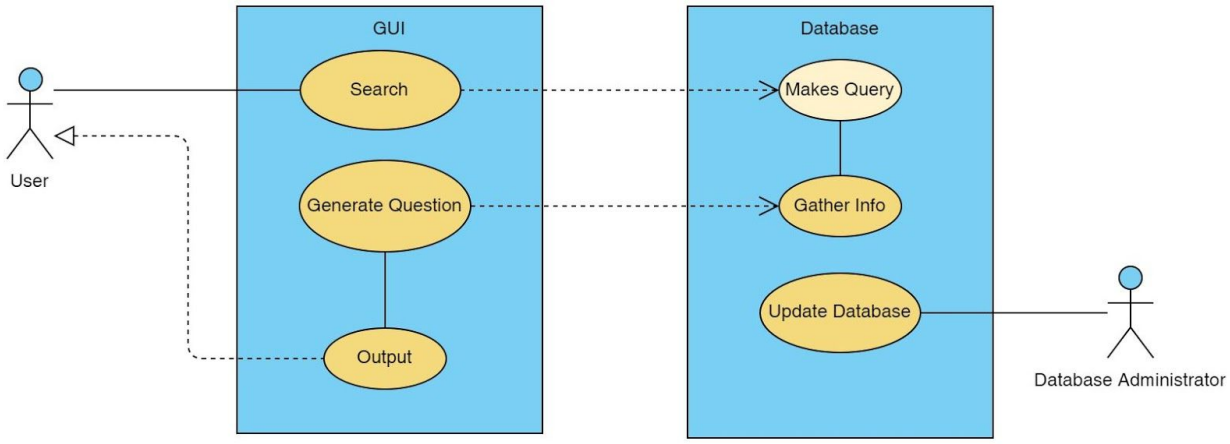
- Risk
    - The risk associated with having a crew child class is having to manage the data flowing between the child and keeping both of the children separate.

Model



In the system we will have two main data types: people and movie. These will each be connected to each other due to movies having many people and people having many movies. Then the people will be split up into either a cast member or a crew member due to cast having a role and crew having a job.

Interaction



In the interaction diagram, the user is on the far left. The user will input a command into the search, which will in turn make a query, gather info that generates a question, which finally outputs back to the user. There are the two interfaces, the GUI and the database due to separate interactions through each other. There will be another type of user, the database administrator that is able to update the information in the database. This distinction is important because the user themself should not be able to update the database.

Section 4: Expectations

Benefits

Creating this system will allow the Department of Film and Media to propagate film media information to students in a format that will greatly aid in the learning outcomes of their introductory course. It will allow the department to train some of the best and brightest film and media students in American culture today.

In the details of our implementation, we garner great benefit from having a movie class is to represent a film and to encapsulate people involved in the film. Furthermore, the abstract implementation of people will greatly simplify searching and allow much of the background implementation to be shared between the various types of people

The benefit of having the child class of crew is that it can have its own data value known as job, since they do not act in the movie: they do the outside jobs such as editing. This benefit also extends likewise to the actor class as they have a particular character (denoted by role as seen in our UML diagram), a feature not shared by crew members.

Assumptions

For this application, a “movie” will be assumed to have had one or more actors, a director, and one or more crew members. It is assumed that people are able to be involved in more than a single movie. We will assume that our users will be students in the Department of Film and Media introductory class. We will assume that our application must be small enough to run on a typical laptop computer within a reasonable response time.

Risks

When it comes to risks, the main risks created are through the layout of classes and the transfer of data between those classes. The risk of the main class of movies is slim to none. This is due to there being very little transfer between except for holding a list of people

However, when it comes to using the abstract class and child class, it could get more complicated. The abstract class could overcomplicate the project, but it would simply be defining each type of person as their own class: the crew or the cast. When managing the data flow and keeping these two objects distinct it could get a little tricky. However, in the long run, it will keep the code cleaner.

Issues

One of the issues that we expect to run into is the relative lack of familiarity our team has with grammar parsing. Since we will be utilizing ANTLR4 to create a grammar for use on this application, this will likely become a difficult technical hurdle. An additional issue that could crop up is related to the edge case of a person that is both an actor and a director, either for the same movie or for different movies. This will have to be handled very carefully so as not to corrupt any data.