Specifications

Peyton Vanhook

Sean Hopkins

Peyton White

Messages and data that will be sent between components

- Login System on Login
    - Sent:
        - What?
            - Username
            - Password
        - Specification:

          {
          “username” : “”,
          “password” :””
          }
    - Receive:
        - What?
            - Boolean for success
            - User ID
        - Specification:

          // If success:
          {
          “success” : true,
          “user_id” : int
          “username” : “”,
          “firstname” :””,
          “lastname” : “”,
          “birthday” : “”,
          “picturepath” : “”,
          “email” : “”,
          }

          // If not successful
          {
          “success” : false,
          }
- User Class:
    - User ID
    - Profile_Picture_Path
    - Username
    - Password
    - Email
    - First and Last Name
    - Birthday
    - User Rated List

    - User Saved List
- Registration Screen on registration
    - Sent:
        - What?
            - Username
            - Password
            - Email
            - Birthday
            - First name
            - Last name
        - Specification:
          Parameters sent in POST request.
    - Receive:
        - What?
            - User Class
        - Specification:
          //if username is not taken
          {
            “taken” : true
          }

- Media JSON Objects:
    - Movie Object

```
“Items”
{
    “0”
    {
        “Id”:int,
        “title”:””,
        “date_released”:””,
        “media_type”: int,
        “picture_path”:””,
        “duration”:””,
        “producers”:{””},
        “actors”:””
    }
}
```

    - Book Object

```
“Items”
{
    “0”
    {
        “Id”:int,
        “title”:””,
```

“date_released”:””,
“media_type”: int,
“picture_path”:””,
“author”:{””},
“num_pages”:””
}
}

- VGame Object

“Items”
{
“0”
{
“Id”:int,
“title”:””,
“date_released”:””,
“media_type”: int,
“picture_path”:””,
“console”:””
}
}

- Media Test Screen on first load
  - Receive:
    - What?
      - 20 (?) Titles of Media
        - Picture
        - Title Name
    - Specification:
      //Array of media items
      JSON Media object based on what category they are in(book,VG,movie).
- Media Test Screen on each submission
  - Sent:
    - What?
      - Rating
        - 1 - 5
      - Title is added to users Rated List
    - Specification:
      Rating is sent as parameter to a POST request.
- Profile Screen on loading list view
  - Sent:
    - What?
      - User_id
        - Request for specified list
          - Saved

- Rated
    - Specification:
      Sent as a POST request with UID as parameter
- Receive:
    - What?
        - Specified list under Users Profile
            - Saved List
            - Rated List
    - Specification:
      //Array of media items
      JSON Media object based on what category they are in(book,VG,movie).
- Home Screen on load:
    - Sent:
        - What?
            - User ID
        - Specification:
          Sent as POST request with UID in parameter
    - Receive:
        - What?
            - Genre list from recommendation Algorithm
            - Media lists for each genre from recommendation algorithm
        - In what format?
        - Specification:
          //Array of media items
          JSON Media object based on what category they are in(book,VG,movie).
- Basic Search on submission:
    - Sent:
        - What?
            - String Searched
        - Specification:
          Sent as POST with string as parameter.
    - Receive:
        - What?
            - Number of returned titles
                - Titles returned as media objects, using media pull function
        - Specification:
          //Array of media items
          JSON Media object based on what category they are in(book,VG,movie).
- Advanced Search on Submission:
    - Sent:
        - What?

            - Title String
            - Author/Cast/Crew/Production Company
            - Date Range (two dates)
            - Genre(s)
            - Keyword(s)
            - Media Type
        - Specification:
          Sent as post request with parameters.
    - Receive:
        - What?
            - Number of returned titles
                - Titles returned as media objects, using media pull function
        - Specification:
          //Array of media items
          JSON Media object based on what category they are in(book,VG,movie).
- Media Item Screen on load:
    - Sent:
        - What?
            - Media ID
        - Specification:
          Sent as POST request with media Id as parameter.
    - Receive:
        - What?
            - Specified Media Information
                - Title
                - Picture_Path
                - Genre(s)
                - Description
                - Rating
            - Will subsequently called recommendation algorithm to pull 5 similar titles.
        - Specification:
          //Array of media items
          JSON Media object based on what category they are in(book,VG,movie).
- Media Item Screen on rating
    - Sent:
        - What?
            - User_id
            - User rating of Title saved to database
                - Title saved to users specific Rated List
        - Specification:
          Sent as POST request with User ID and rating as parameters.

    - Receive:
        - Confirmation as json object with true/false.
- Media Item Screen on save
    - Sent:
        - What?
            - User_id
        - In what format?
        - Specification:
          Sent as POST request with user ID as parameter.
    - Receive:
        - Confirmation as json object with true/false parameter.
- Pull Rated List
    - Sent:
        - What?
            - User_id
        - Specification:
          Sent as POST request with user id as parameter.
    - Receive:
        - What?
            - JSON object based on media type(book,movie,VG)
        - Specification:
          //Array of media items
          JSON Media object based on what category they are in(book,VG,movie).

  Pull Saved List
    - Sent:
        - What?
            - User_id
        - Specification:
          Sent as POST request with user id as parameter.
    - Receive:
        - What?
            - JSON object based on media type(book,movie,VG)
        - Specification:
          //Array of media items
          JSON Media object based on what category they are in(book,VG,movie).
- Recommender Call (Media ID’s)
    - Sent:
        - What?
            - Array of media ID’s
            - Max num of items returned
            - Min association value of items returned
        - In what format?

        - Specification:
    - Receive:
        - What?
            - Array of pairs, media ID's with association value.
        - Specification:

          {

          "media_ID": 0,

          "Similarity": 0.0

          }
- Recommender Call (Genres and Keywords)
    - Sent:
        - What?
            - Array of genres
            - Array of keywords
            - Max num of items returned
            - Min association value of items returned
        - In what format?
        - Specification:
    - Receive:
        - What?
            - Array of pairs, media ID's with association value.
        - Specification:

          {

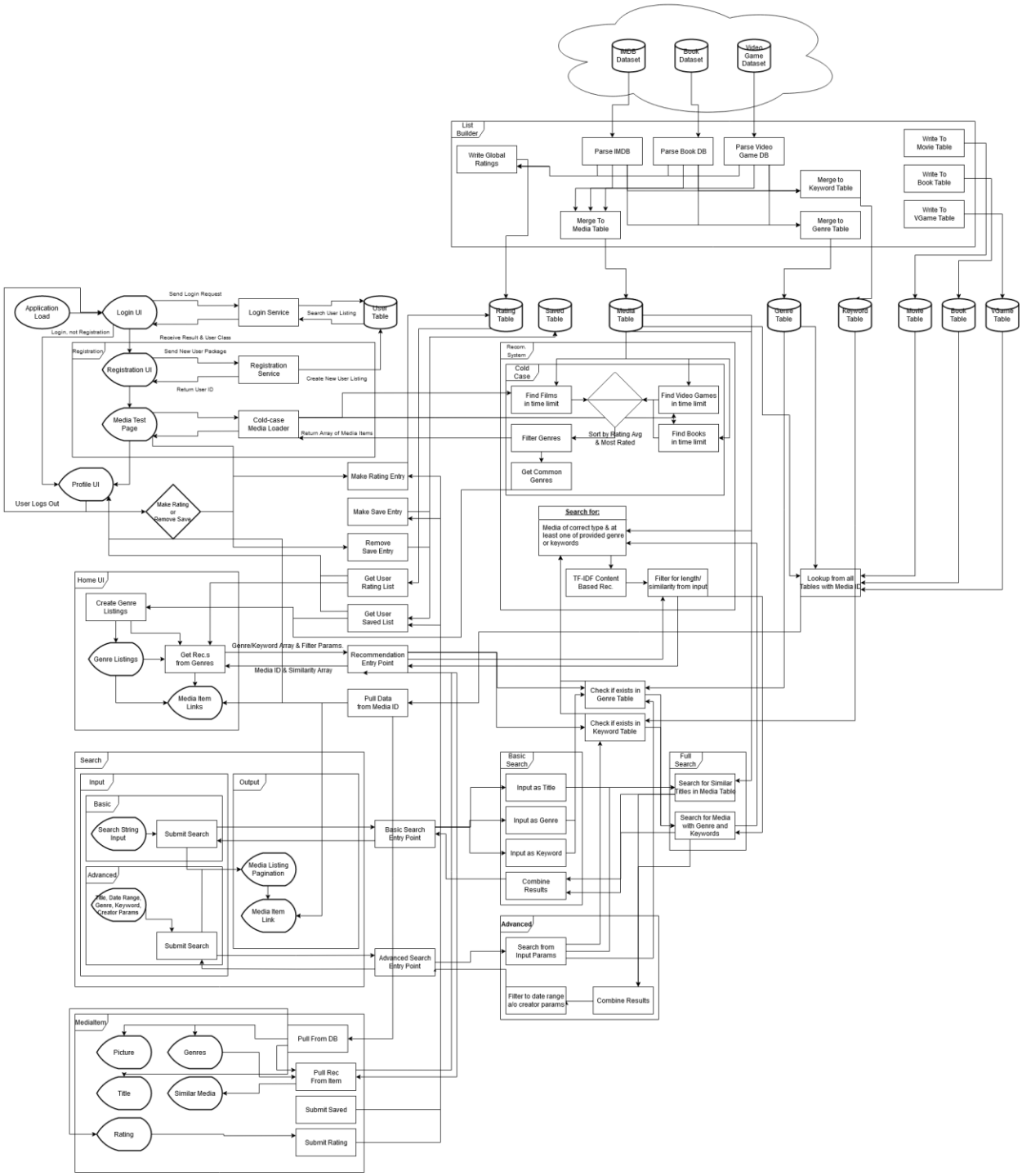
          "media_ID": 0,

          "Similarity": 0.0

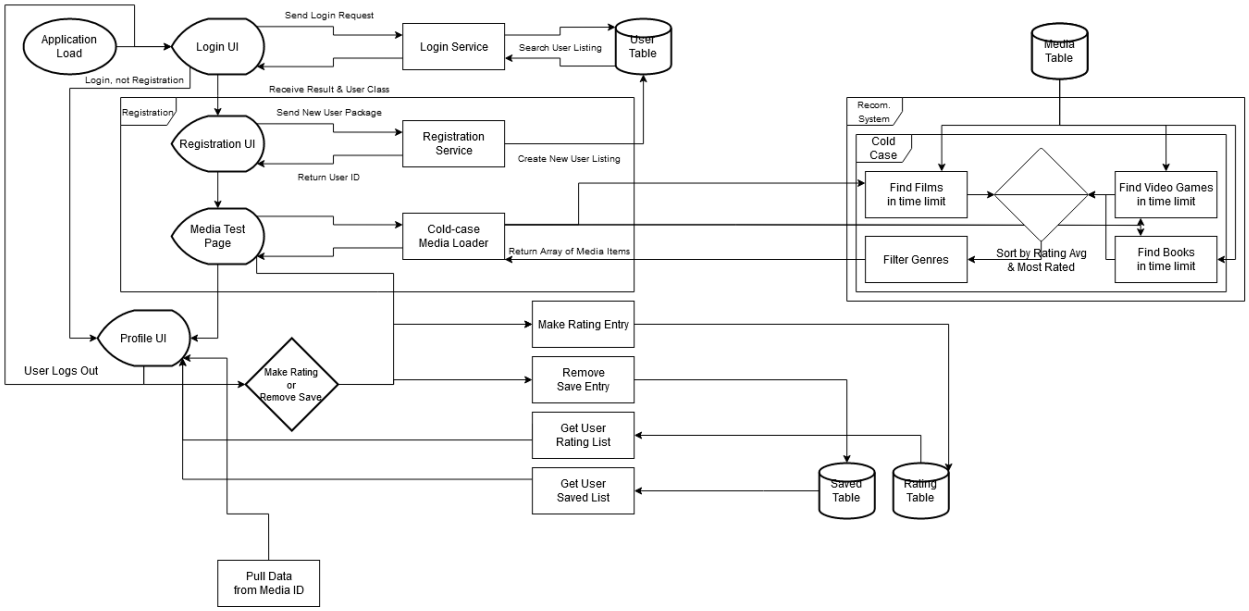
          }

Block Diagrams

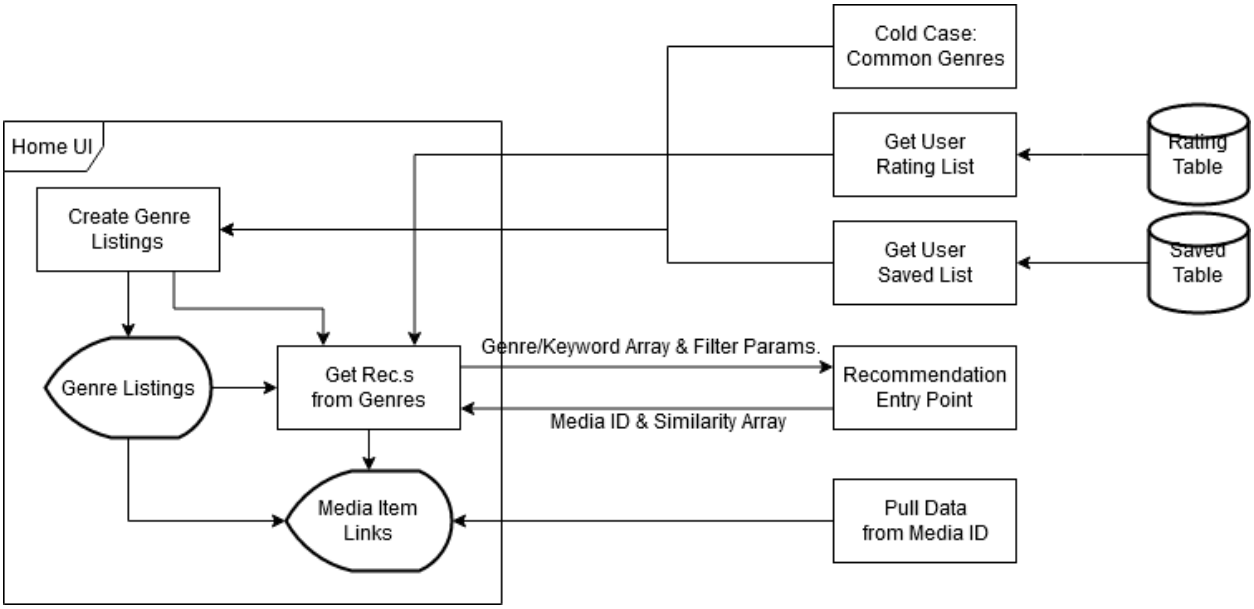
Full block diagram


IMDB Dataset
Book Dataset
Video Game Dataset
List Builder
Write Global Ratings
Parse IMDB
Parse Book DB
Parse Video Game DB
Write To Movie Table
Write To Book Table
Write To VGame Table
Merge to Keyword Table
Merge To Media Table
Merge to Genre Table
Application Load
Login UI
Send Login Request
Login Service
Search User Listing
User Table
Login, not Registration
Receive Result & User Class
Registration
Registration UI
Send New User Package
Registration Service
Create New User Listing
Return User ID
Media Test Page
Cold-case Media Loader
Return Array of Media Items
Profile UI
User Logs Out
Make Rating or Remove Save
Rating Table
Saved Table
Media Table
Genre Table
Keyword Table
Movie Table
Book Table
VGame Table
Recom. System
Cold Case
Find Films in time limit
Find Video Games in time limit
Sort by Rating Avg & Most Rated
Filter Genres
Find Books in time limit
Get Common Genres
Make Rating Entry
Make Save Entry
Remove Save Entry
Get User Rating List
Get User Saved List
Search for:
Media of correct type & at least one of provided genre or keywords
TF-IDF Content Based Rec.
Filter for length/ similarity from input
Lookup from all Tables with Media ID
Home UI
Create Genre Listings
Genre Listings
Get Rec.s from Genres
Media Item Links
Genre/Keyword Array & Filter Params.
Media ID & Similarity Array
Recommendation Entry Point
Pull Data from Media ID
Check if exists in Genre Table
Check if exists in Keyword Table
Search
Input
Basic
Search String Input
Submit Search
Advanced
Title, Date Range, Genre, Keyword, Creator Params
Submit Search
Output
Media Listing Pagination
Media Item Link
Basic Search Entry Point
Advanced Search Entry Point
Basic Search
Input as Title
Input as Genre
Input as Keyword
Combine Results
Advanced
Search from Input Params
Filter to date range a/o creator params
Combine Results
Full Search
Search for Similar Titles in Media Table
Search for Media with Genre and Keywords
MediaItem
Picture
Genres
Title
Similar Media
Rating
Pull From DB
Pull Rec From Item
Submit Saved
Submit Rating

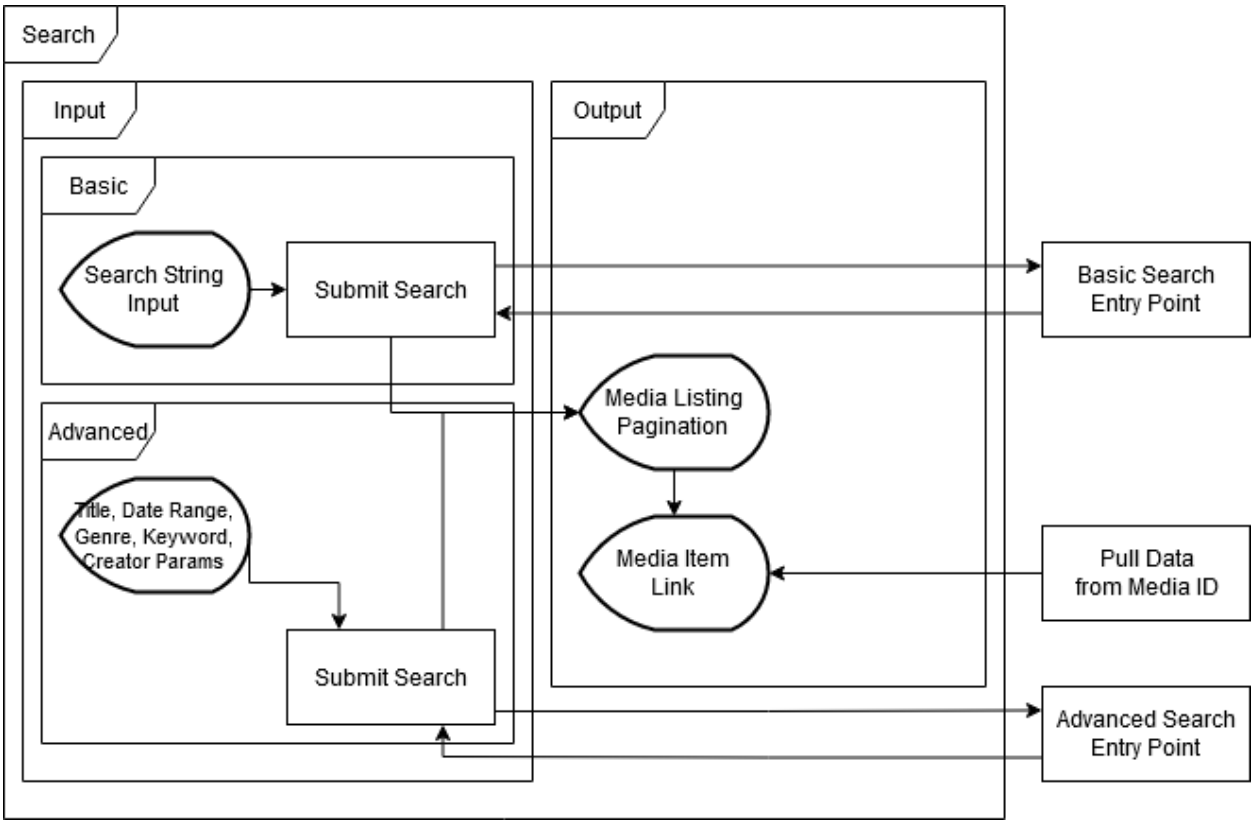
Login/Registration



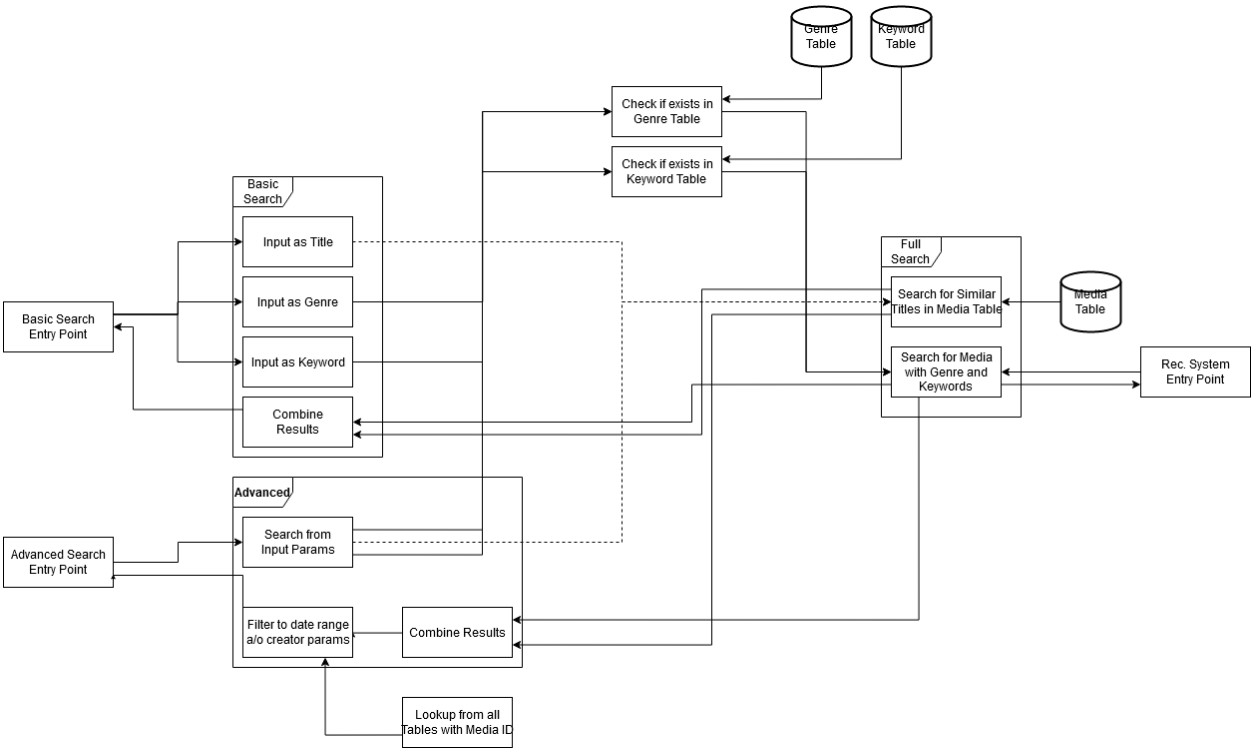
Home



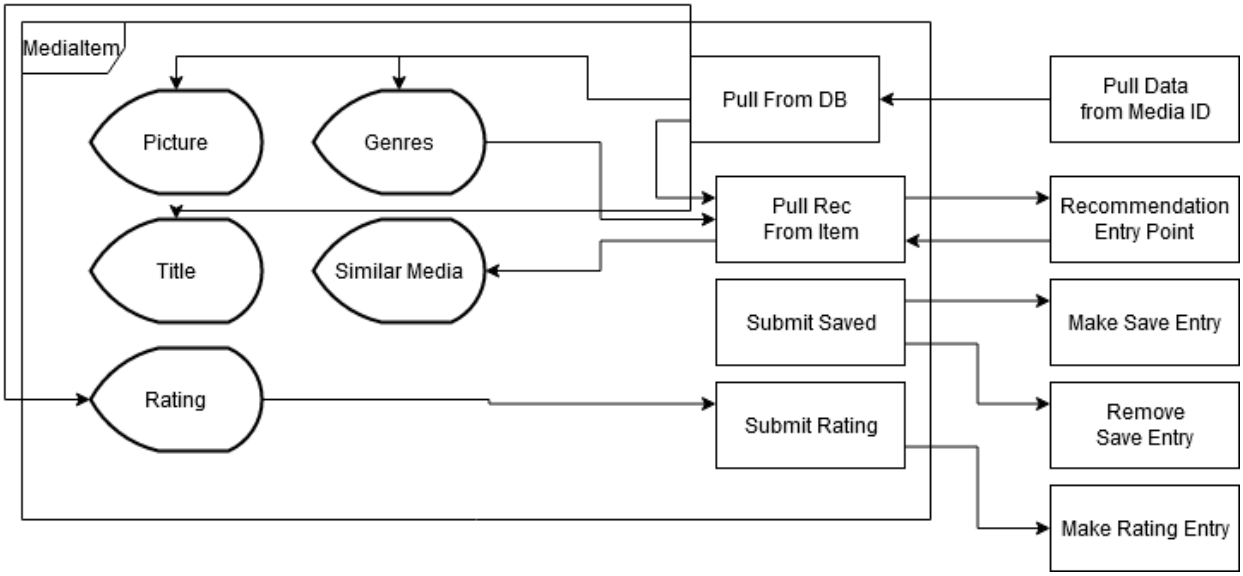
Frontend Search



Backend Search



Media Item



Recommendation System

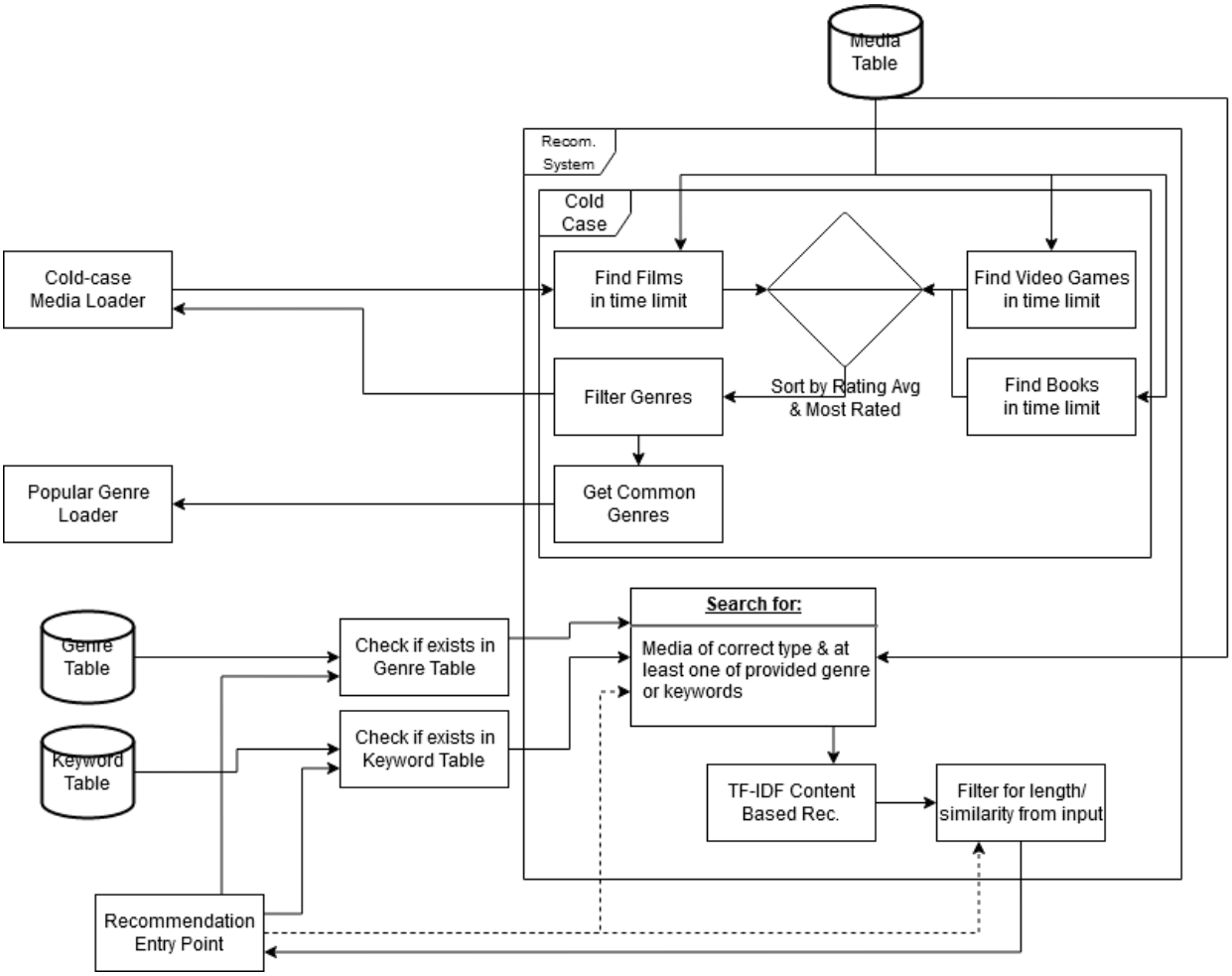
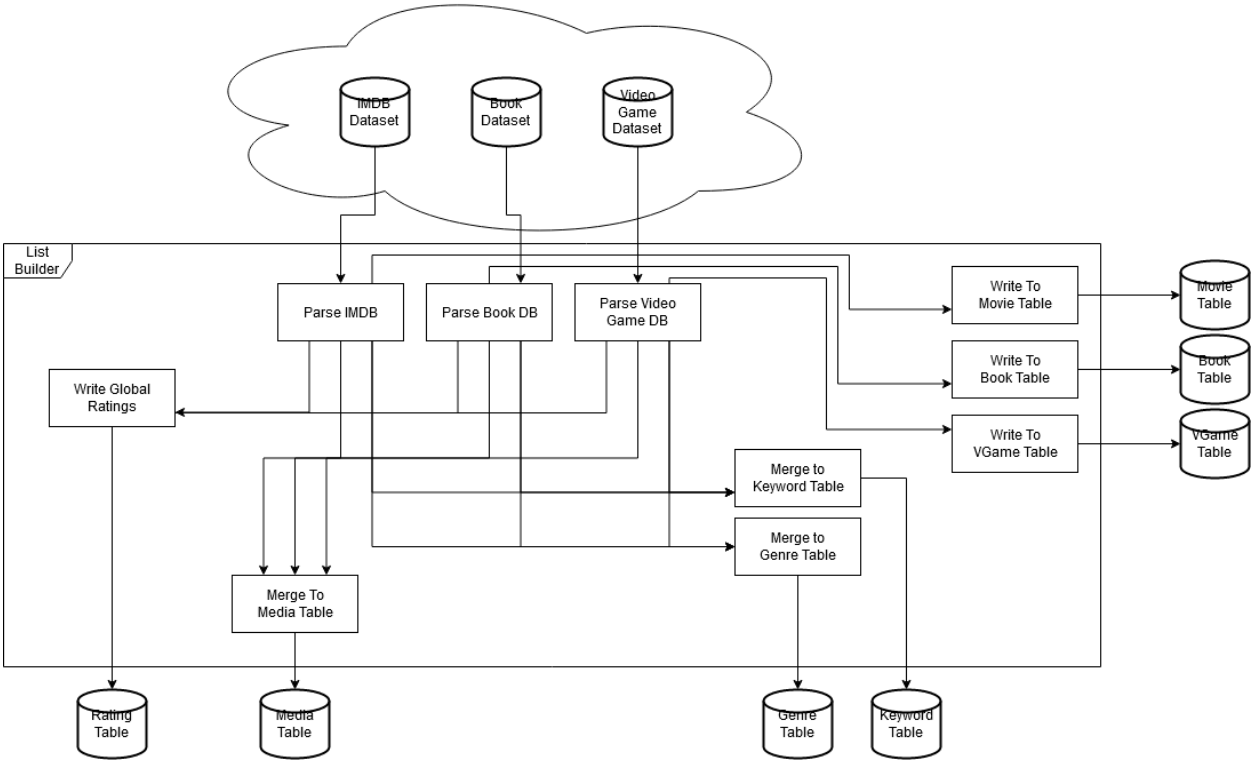
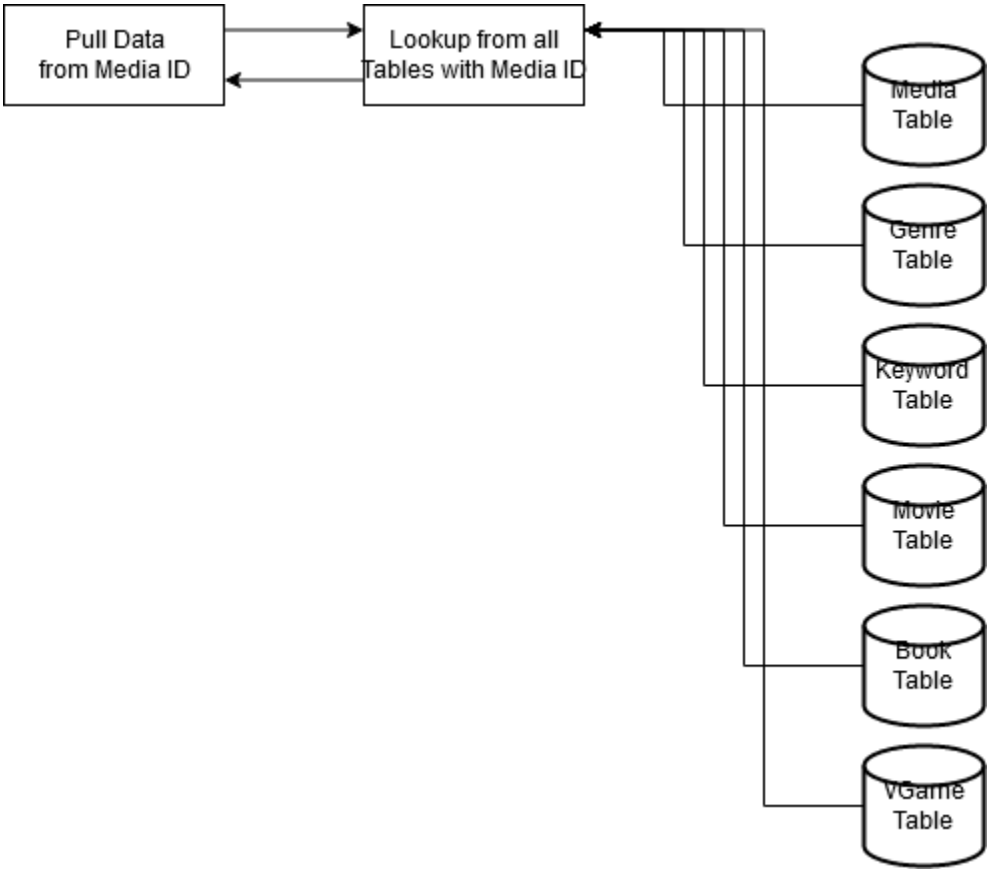


Table Builder

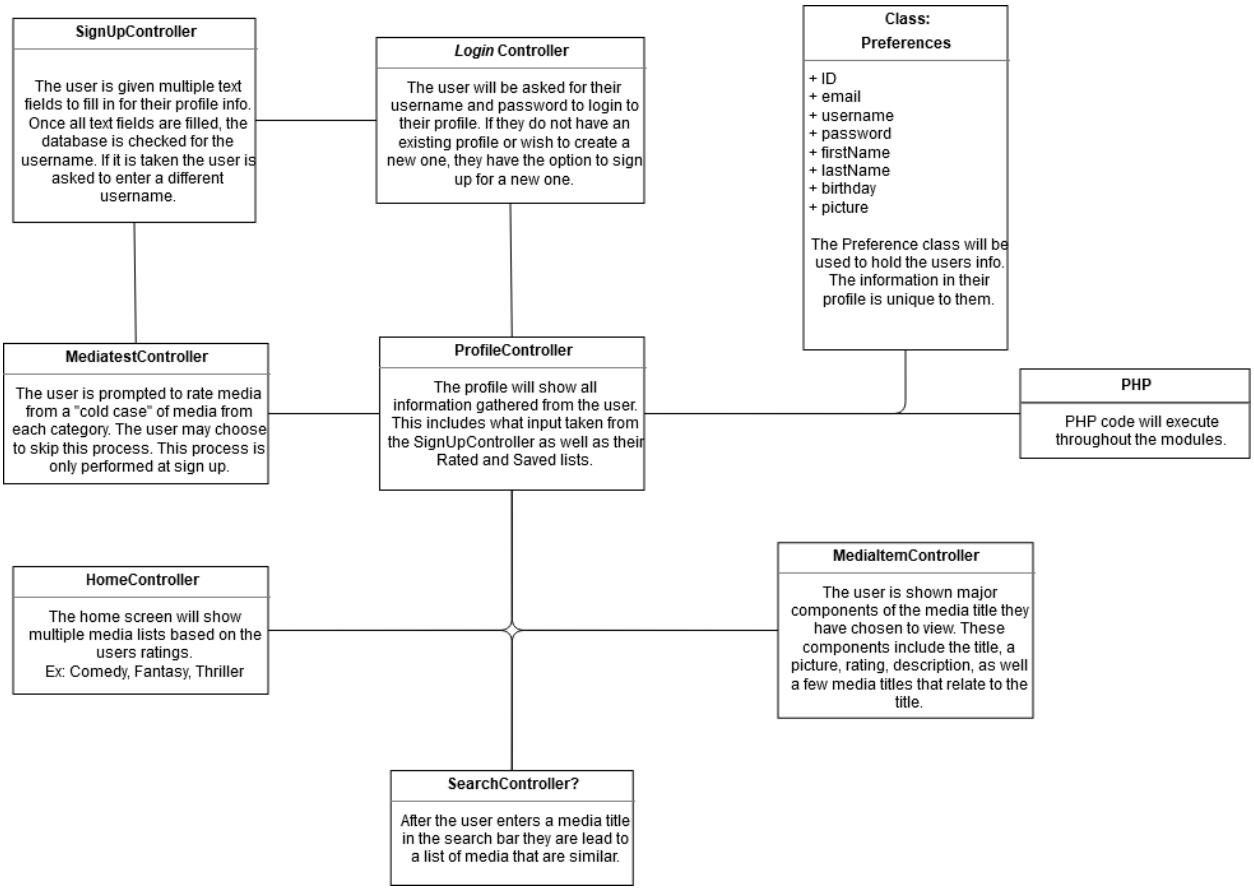

IMDB Dataset
Book Dataset
Video Game Dataset
List Builder
Parse IMDB
Parse Book DB
Parse Video Game DB
Write To Movie Table
Write To Book Table
Write To VGame Table
Movie Table
Book Table
VGame Table
Write Global Ratings
Merge to Keyword Table
Merge to Genre Table
Merge To Media Table
Rating Table
Media Table
Genre Table
Keyword Table

Pull Data from Media Tables

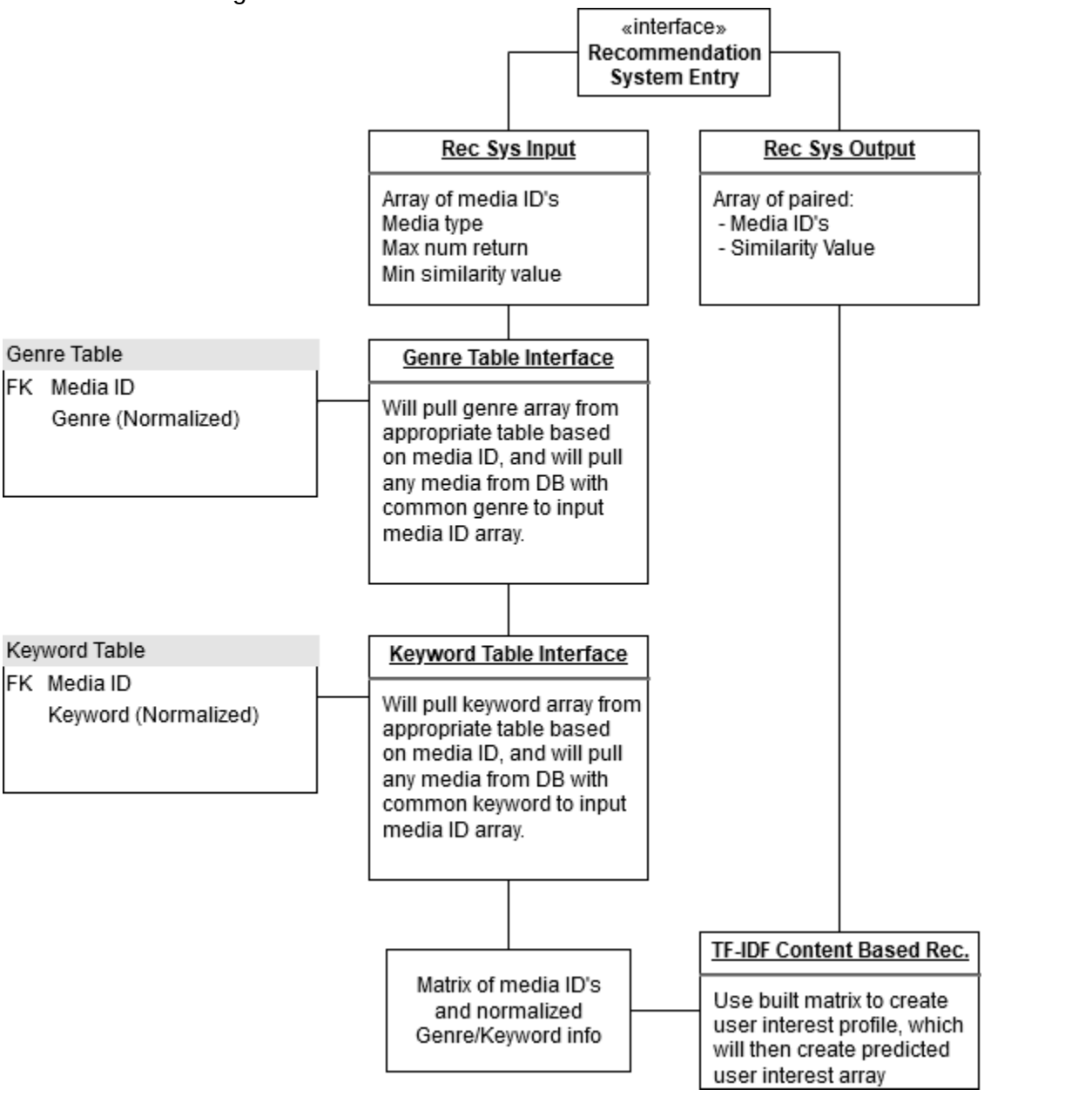


UML Diagrams

Frontend Java:



Recommendation Algorithm:



Media Info Pull:

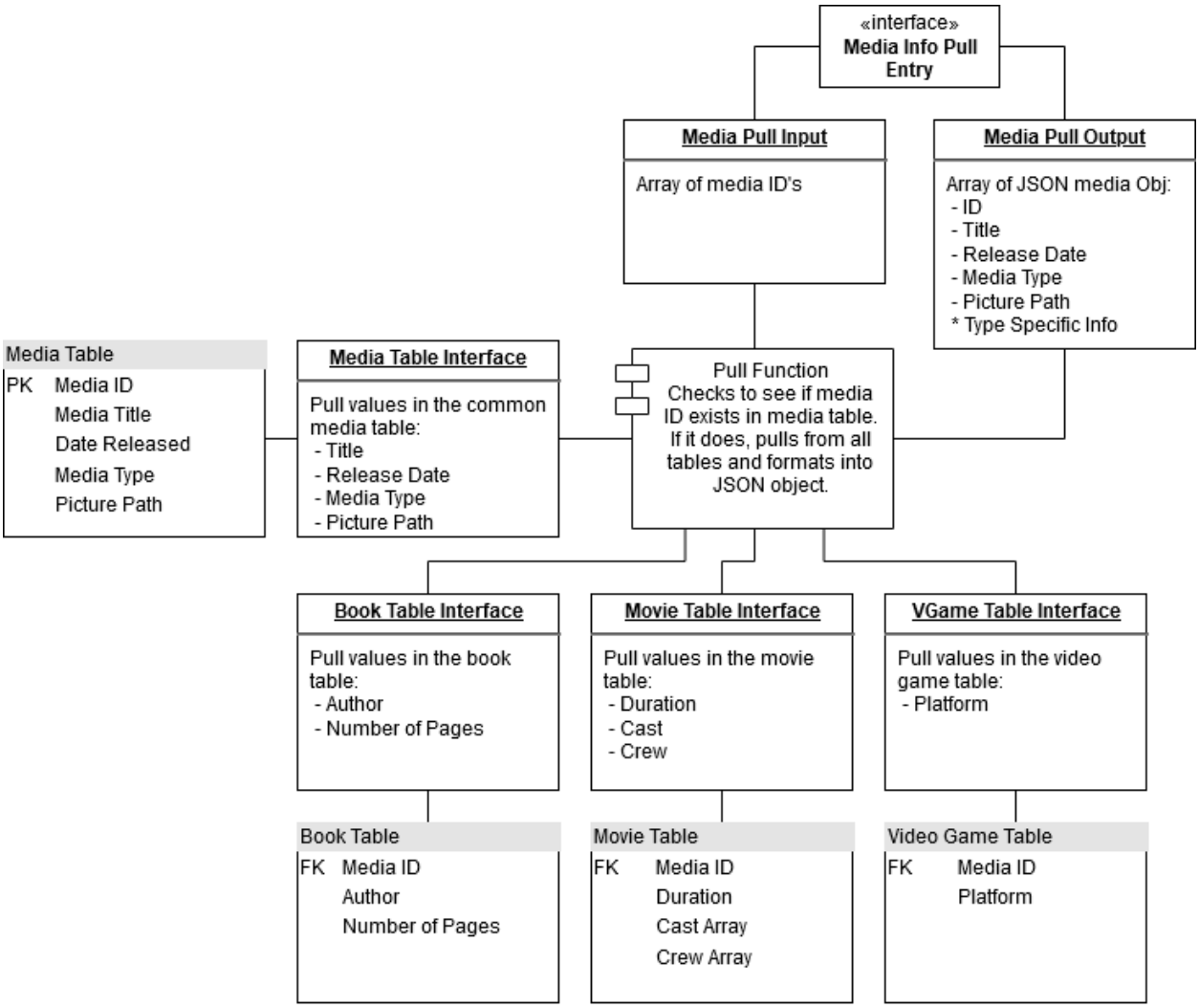
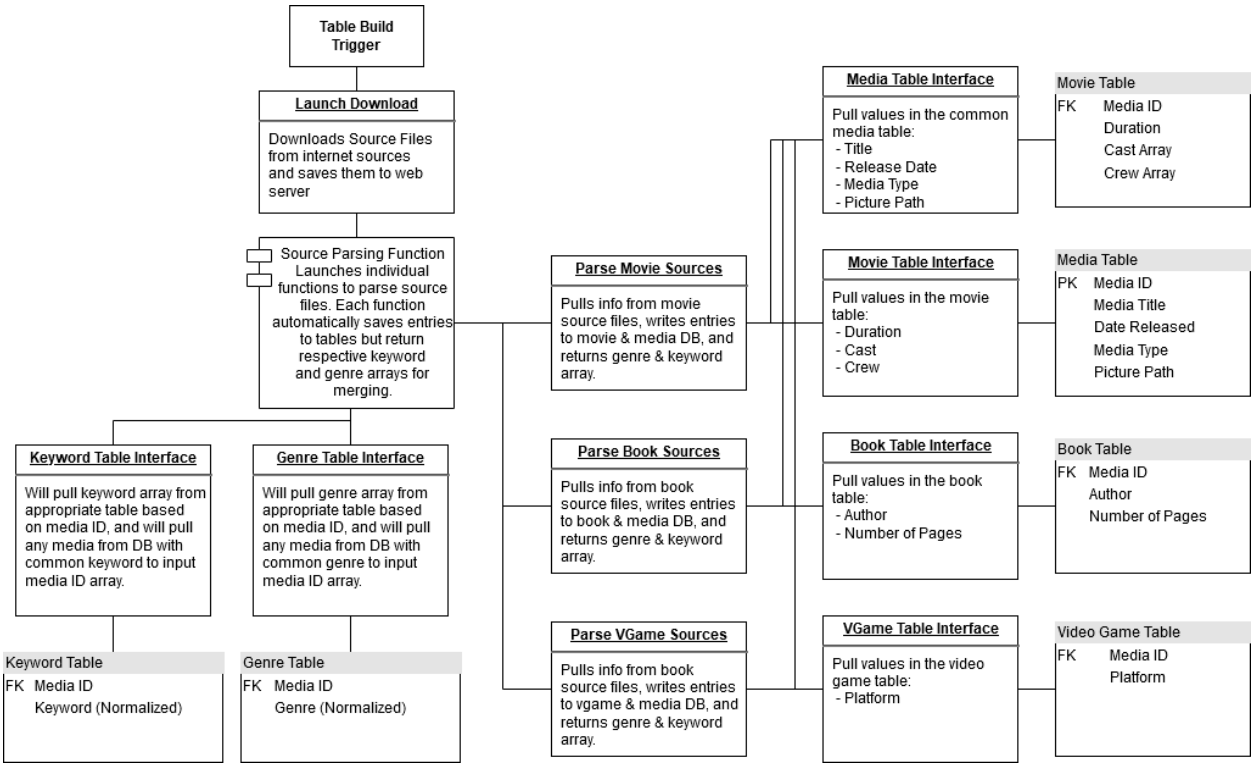
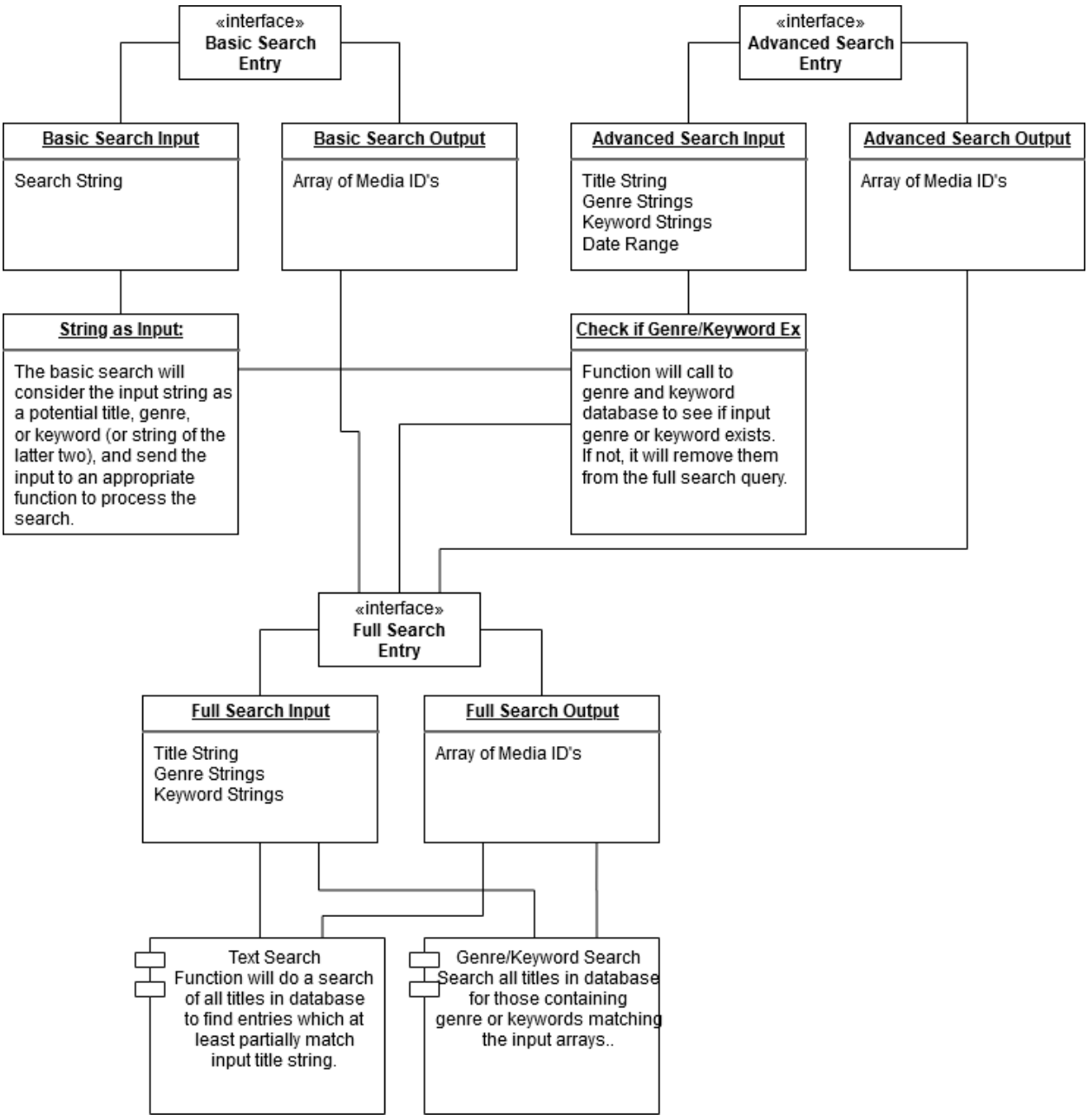


Table Builder:



Search System:



Description

The G.O.A.T. was made for those who have felt the pain of finishing any form of media that left them breathless just to be left with nothing else to keep the rush going. The G.O.A.T. recommender is for those people who have the worst luck with media droughts. The frustration of running out of interesting media is a problem of the past with the G.O.A.T. With a name like this it has to work. The G.O.A.T. provides recommendations for a few different types of media. These include books, movies, and video games. All necessary for the diet of a healthy G.O.A.T.

As a user first gets into the G.O.A.T they are prompted for their username and profile. These two criteria are used to unlock the joy of experiencing a constant stream of user interest-specific media. Once unlocked the user preferences saves all the information about the user from the database for easy access. There are no worries for those first timers as they also have the option of allocating some space for themselves. Once a profile is created, the user information gets sent and saved in the database and they can now use the G.O.A.T. to its highest potential as long as they feed it well! The more you feed that algorithm, the better your chances of finding the perfect media for you. The G.O.A.T craves that mineral for a reason people. Get with the program! While this sounds great, a more in depth description is needed in order to fully understand how the G.O.A.T. actually does what we boast about.

On Login the user is prompted for the usual criteria to keep their profile information safe. In order to unlock their information we need to make sure to set some ground rules. These inputs are used to search through the database of the G.O.A.T. where the users specific information is being held. In short this information gets passed through PHP and checked in the database. What can be found is a treasure trove of information that the G.O.A.T. algorithm uses to create the best recommendations possible. Not only is the typical profile information saved in the users profile, but a list of items rated and saved by the user themselves is hoarded as well. This information is crucial in the act of finding the perfect match for users.

As stated before, a user has the option to create a profile if they do not have one themselves. The process of getting a G.O.A.T. for the first time is no laughing matter as it can be the most crucial time. We make sure to give the user the best options to help their G.O.A.T. get to know them right off the bat. Once the required information for registration is filled, the user is prompted with the Media Tester. Tester anxiety is no big deal here. All we ask is for a bit of information to start the user off well with their new companion. The Media Tester is optional, so the user may move on and ignore this part if they desire. We have found other ways to gather the information needed to find the best recommendations for our users. However this is not recommended as the Media Tester is built to give the user media of all types. That and the G.O.A.T. gets hungry and they will eat literally anything.

Knowing which types of media are liked is just as beneficial to your G.O.A.T. as those that are disliked. A starved G.O.A.T. has nothing to go on. If users expect great things, they must feed their G.O.A.T. well. The reason that feeding the G.O.A.T. is so important is because of the way that it is designed. Without information to go off of, there is only so much that can be done. Therefore, G.O.A.T. uses the ratings of its user to narrow down its results and really get a taste for the media that will best suit them.

Whether the Media Test is administered or not, the user is given a User ID specific to their username and then sent to the Profile screen. On the Profile, the user's information is displayed. Users will have the option to choose a picture for their G.O.A.T. that is saved on their desktop as well as display user specific information like their name, birthday and so on. Another feature of the Profile is the display of the users specific Rated and Saved lists which is saved in the database when the user ranks or saves a media item. As previously stated the Rated list is used by the G.O.A.T. to get a feel for the users preferences and provide the best recommendation we can provide. The Saved list is a neat feature for the user to make sure that any interesting media is not lost in the vast pile of media options. These lists are crucial to the user so displaying them in their profile is where we thought they ought to be.

While having a profile for the user is good, the G.O.A.T. needs somewhere to digest the information the user gives it and display it for them. The Home screen is where that happens. At Home a G.O.A.T. takes hints from the information and provides lists based on those media that are most prevalent among their users highest rated media. Media on the home screen is listed by certain columns from our database; that include genres, keywords, or media titles. What is displayed in these lists is similar to that of the Rated and Saved lists, but their criteria is based on the “title” of that list. This is the best spot for those who properly feed their G.O.A.T. a robust, healthy diet of media ratings.

The Home screen is one of the two places that the G.O.A.T. may recommend new items for the user. The Search List is the second. A user has the ability to give their G.O.A.T. searches for specific media titles in order to find media that most resembles them. The search tool is useful for those who know what they want but are having trouble finding media similar to experience. The first search option is the basic search. The user can simply type a title and press search and the G.O.A.T. is off to eat. With that title, the database will get searched through millions of titles for that certain media. The end result will be anything that has that title in it. From there the user can select which one they were thinking of and save it, or rate it. The next search is the advanced search. From here the G.O.A.T. is hungry for a very specific type of food. The user will search a title however, based on the media title the user will be required to fill out other fields that involve the author, producers, date released, etc…. From here the database will get searched thoroughly and it will retrieve that one specific media the G.O.A.T was hungry for. They can then do the same thing, save, or rate this media.

One might wonder what happens once the G.O.A.T. produces a media that catches the users eye. A user needs some way of getting more information than the basic picture and title. This is where the Media Item screen comes into play. When a media title or picture is clicked the user is directed to the Media Item screen where they are provided with all of the information that the G.O.A.T. database provides. The G.O.A.T. is a great multitasker and knows what the

user wants so the information does not stop there. On display is a short but concise list of the top five media related to that title. The G.O.A.T. is relentless in its work to do its job. The user is also given the chance to rate the media title.

In all, the G.O.A.T. is useful in its ability to keep the entertainment going. It works best when the user gives it information to work off of and it uses user specific data to find the best recommendations for them. Those of us developing the G.O.A.T. hope users have a great experience and find exactly what they are looking for. Now go on and try it out. Returns are also not an option.