http://flip3.engr.oregonstate.edu:60075/ Cumberworth/Narayanan
[Use VPN if off-campus]

CS340 Project Step 6: Draft UPDATE and DELETE

Sriram Narayanan: Backend/Integration with Frontend contributions
Mike Cumberworth: Fullstack

UPDATES for STEP 6

CRUD setup for user, media items linked with genres.
Item_genres and genres are working as well

Feedback from Peer Review & Instructors from STEP 5:

Aidan Fleischer

Would it be possible to add error that redirects the user when a user clicks on an "Add Item +" prompt? If there's a missing param you can redirect via some middleware in the routing process (I'm partial to https://express-validator.github.io/docs/). That might provide a smoother flow, or alternatively you could hide the header options that aren't selectable before a user has logged in. Keep it up!

Feedback from Peer Review & Instructors:

Kevin Hill

I like the aesthetic and colors on the UI. Also based on the schema it looks like you database design fulfills all the requirements.

I would make sure that the requirement to make one relationship NULLable. Like for instance Genre with media item. I think this would be an optional relationship due to the fact your search goes off of the title.

Overall, I think it is a solid UI design.

William Wells

I could actually use this IRL. :)

Curious about the "I'm Feeling Lucky" button... does it pick a random item from your queue? I only find it a little confusing because it's next to the Add button. Regardless, great work!

Alessio Peterson

I really like this idea. I could definitely use a method to track all the books and movies I want to watch. It seems like you hit everything on the checklist and your UI is easy on the eyes. Great job!

Michael Gabriel

Hi Mike and Sriram,

I love the webpage. I looks clean and well thought out. I can't see anything on here that looks like it falls short of the requirement, great work.

Mae Lapresta (TA)

I really like your webpage, and your navbar looks great! I really love how you implemented the M-to-M relationship on the insertion of new media items, with a genre selection. Right now your site is missing a way to add genres, and a delete from your many to many relationship.

Actions Based on Feedback From Peer Review:

All peer reviewers liked the UI, describing it as "solid", "easy on the eyes" and "clean and well thought out". Also the implementation of the M-to-M relationship on the insertion of new media items, with a genre selection was looked at favorably.

One reviewer reminded us that one relationship should be NULLable and gave an example of Genre with media item. Genres, Original Language Title, and Publication Year are all optional values that can be left NULL, so this should be covered.

One reviewer was curious about the "I'm Feeling Lucky" button and confused about its function. This has been removed.

Reviewing TA pointed out that the site is missing a way to add genres and a delete from a many-to-many relationship. Added buttons for handling multiple Media Queues as well as input and button to add a new Genre. Multiple media queues will be the many-to-many where we allow for deletion. We will not allow for deleting a genre or item_genre, as that would impact all users who've added that item to their queues.

http://flip3.engr.oregonstate.edu:60075/ Cumberworth/Narayanan
[Use VPN if off-campus]

Updates to Project or Database Outlines

Search capability added which calls the iTunes Search API and displays the top 5 results. The search input is accompanied by a dropdown which allows the user to choose the media type. The user will be able to choose an entry to add to the media queue. This search & filter will also tie in w/local DB items.

Moved the single "Add" button from Search into the results table with an ID for the item's index, so clicking it can kick-off adding that individual item to the database.

The “Project Outline” and “Database Outline” sections below have been updated to reflect all these design changes.

http://flip3.engr.oregonstate.edu:60075/ Cumberworth/Narayanan
[Use VPN if off-campus]

Fixes based on Feedback from Step 1:

Changed all “string” data types to varchar with appropriate length restrictions.

The other suggestion was to make **media_type** an attribute of the **media_item** instead of a separate entity. We had noted that as a possibility. One possible hiccup with this approach is if the **media_item’s** “type” is a “collection” of other media_items. Not sure if that may cause issues. For now that suggested change has been made, replacing the media_type Entity with an ENUM of possible values. This leaves us with five Entities.

Project Outline and Database Outline - Updated Version:

Outline (last updated 4 Nov)

Our proposed project is based on the idea of “Tsundoku”（積読 a Japanese term for the practice of buying books but never quite getting around to reading them. This project aims to help you catalog and track your library of “someday” books, movies, and other media.

Most of us have some “classics” we mean to read, or a pile (virtual or hardcopy) of programming tutorial tomes you don’t have time for right now. Maybe you prefer visual media, and *eventually* plan to binge every season of “Game of Thrones,” or that entire collection of Kurosawa films you will get around to “some day.” This web app will handle those too, along with any audio albums you’ll maybe listen to.

You will start out by creating a User “account” with your name and email address.

Next, add a new media queue to hold items you’ve bought and might at some point look at - because that is the defining factor of Tsundoku. A media queue can be a reading list, watchlist, or playlist that you will fill with “queue_items” pulled from the database.

Each “media item” in the database has an English title, optional foreign/original title, a “media type,” (book, film, etc), a list of genres, and an average user rating if available.

An item added to a queue becomes a “queue_item.” It has a “status” as having been started or not, “date added” so you can see how long you’ve procrastinated, and a “priority” toggle that will bump that item to the top of your list when sorted appropriately. Whenever you finally get around to an entry, you can mark it complete and/or delete it from your queue.

A user may also now have multiple queues (to facilitate the required cascade deletion from a many-to-many relationship).

http://flip3.engr.oregonstate.edu:60075/ Cumberworth/Narayanan
[Use VPN if off-campus]

Database Outline (last updated 4 Nov)

The following covers the Entities, Relationships, most data types, attribute names, and some restrictions where they make sense. It generally follows the SQL Styles found here: https://www.sqlstyle.guide/ , with lower-snake_case entity, table, and column names, plural table names, and singular everything else.

Entities

user

- user_id, an auto-incrementing, non-null, int (PK)
- username, non-null varchar(100)
- user_email varchar(100)

media_queue

- media_queue_id, auto-incrementing, non-null int (PK)
- user_id, the int id of the user that owns the queue, non-null (FK)

queue_item

Entity for once a media_item has been added to a media_queue (since items may exist that aren't currently in any queues).

- queue_item_id, int auto-increment, non-null, PK
- status: ENUM "0=NotStarted" "1=InProgress" "2=Complete", defaults to 0
- priority BOOL (rather than messing w/numerical ordering, it either is or isn't a priority)
- date_added DATETIME non-null, set automatically on insert
- media_queue_id, int, non-null FK
- media_item_id, int, non-null FK

media_item

- media_item_id, a non-null, auto-incrementing int (PK)
- media_type, non-null, ENUM
    - 0=book, article, periodical, tv_show, film, short, collection, audio_album
        - "book" covers print & all ebooks;
        - "audio_album" includes all genres
        - "collection" indicates the media_item itself contains other media items. Not sure yet if/how well this sort of recursive relationship will work.
- title, non-null varchar(255),
    - Optional non-English/original title (if UTF-8 text works properly), varchar(255)

- publication_year as nullable int
- Overall “star” rating of the item, nullable int from 0-5.

genre

- genre_id (auto-incrementing, non-null, int, PK)
- genre_name

Relationships

- **user_id** to **media_queue is** one-to-many; a user may now have multiple queues, each with multiple items

- **media_queue** to **media_item** is many-to-many; many queues may contain a media_item, and a media_item may be added to multiple queues

    - This M:M relationship is facilitated via the “queue_item” Entity and table

    - Deletion of a queue_item, or clearing/deletion of an entire media_queue will count for our deletion from an M:M relationship. We will not be deleting ‘genres’.

- **media_queue** to **queue_item** is one-to-many; A queue contains multiple items
    - A given item should only be able to be added to a queue once, but can be in multiple queues under the same user.

- **media_item** to **queue_item** is one-to-many, because a single media item could be part of many user queues.

- **media_item** to **genre** is many-to-many, for which we will have the **item_genre** table

- **item_genre (many-to-many relationship)**
  Junction table since a media_item can have many (sub)genres, and a genre can apply to many media_items. As long as item_genre doesn’t add any additional fields, we can get away with using the two FK’s as a composite Primary Key for this table[1,2], and have removed the need for an additional integer PK.

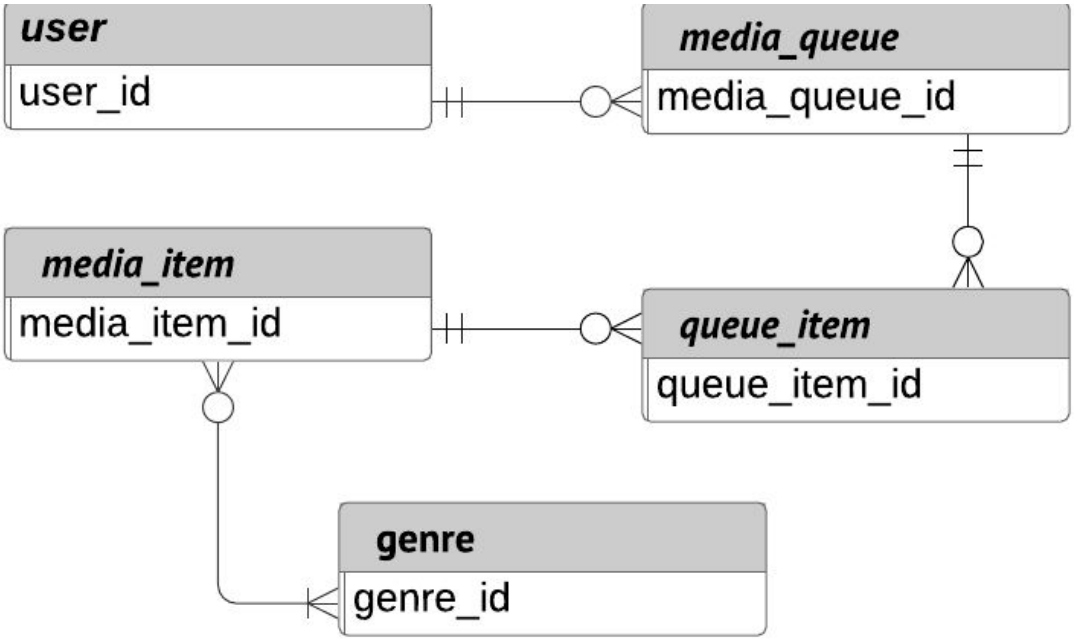
    - media_item_id, non-null, int FK
    - genre_id, non-null, int FK

---

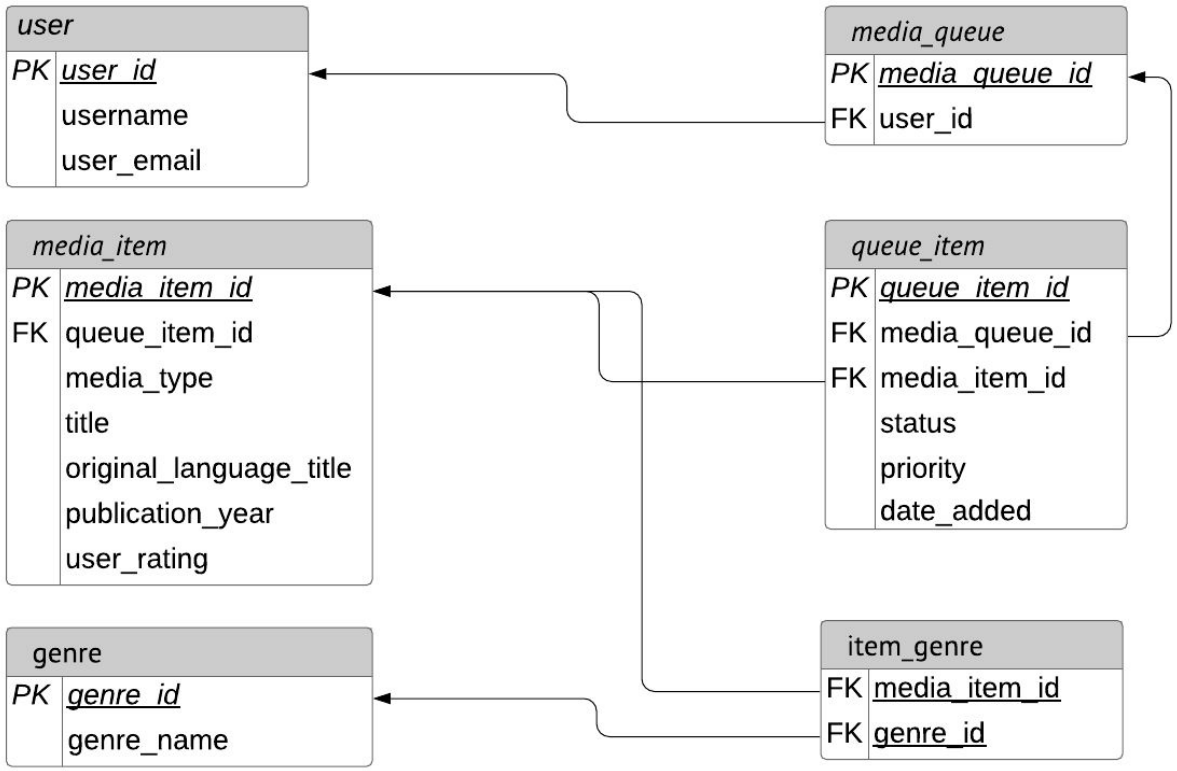
[1] https://stackoverflow.com/questions/790334/sql-do-you-need-an-auto-incremental-primary-key-for-many-many-tables

[2] https://stackoverflow.com/questions/2190272/sql-many-to-many-table-primary-key?noredirect=1&lq=1

c) Entity-Relationship Diagram



d) Schema:


user
PK user_id
username
user_email
media_queue
PK media_queue_id
FK user_id
media_item
PK media_item_id
FK queue_item_id
media_type
title
original_language_title
publication_year
user_rating
queue_item
PK queue_item_id
FK media_queue_id
FK media_item_id
status
priority
date_added
genre
PK genre_id
genre_name
item_genre
FK media_item_id
FK genre_id