Shows & Tel.

A TV show database for CAMS students at Wellesley College.

Catherine Chen, Chloe Moon, Alice Zhou

Overview

Shows & Tel. is an educational database of TV shows that allows TV writing students at Wellesley College to find information about shows for TV screenwriting courses. Currently, the process of finding examples to supplement coursework requires a multi-step search process at the expense of the student. In an effort to streamline this process, we have designed a web interface that allows students to search for TV shows, edit existing ones, and add new profiles for TV shows that are not already in the database. Each show has its own profile page, complete with **basic information like initial release dates, creators, and networks**, and is also equipped with supplementary information such as links to **scripts**. There are also **tags** that allows students to search for shows more specifically. For example, if a student is writing a workplace pilot, they can search that tag and come up with a list of workplace shows. Or cop shows, extended family shows, mockumentaries, etc. In addition to tags, there are also **content warning tags** that both give people who need the warnings and also help students find examples of shows tackling tough subjects. Overall, Shows & Tel. has the potential to be used at a much larger scale by other institutions with similar screenwriting courses, or even the larger entertainment industry in general.

Motivation

The motivation for this project came from seeing the various screenwriting and TV writing classes at Wellesley and ascertaining a need for a database that was specifically created for screenwriters. Currently, databases such as IMDB are catered to the general user, rather than writers themselves. For aspiring writers, it would be incredibly helpful to see different scripts for TV shows and organizing the TV shows based on what they would want to imitate and write. For instance, writers that wanted to write shows on the Asian American experience could see previous shows that also wrote about this topic. Tags could also be based on pace or length, so if they wanted to write fast paced shows, long shows, etc. Besides tags and being able to upload scripts, another feature that was notably missing in current TV databases was content warnings. This could be important for people who want to skip certain shows, or even if a writer wanted to see how past shows have tackled these difficult topics. Because of these lack of features in current databases, we wanted to see if we could fill that niche for aspiring screenwriters at Wellesley, and perhaps beyond. We have been communicating with our client, Professor Lauren Holmes, and actualized the needs to a real project.

User Guide

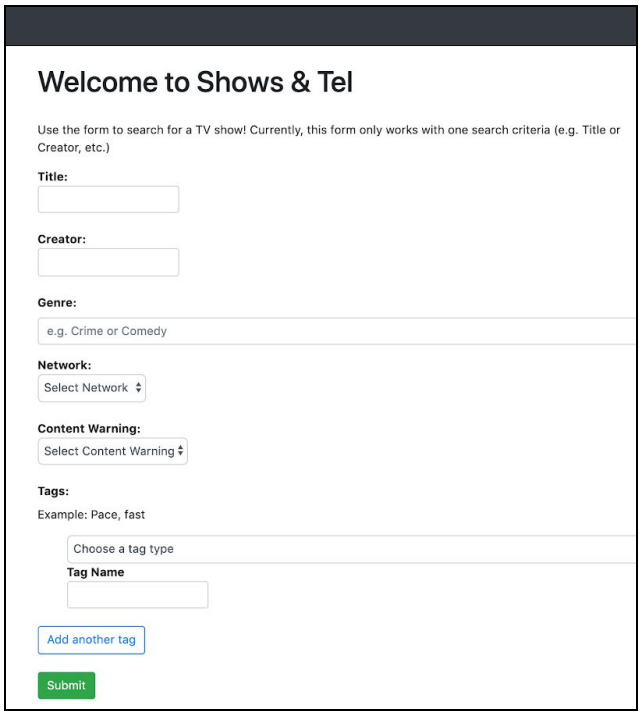
Currently, Shows & Tel. allows two types of users: **A. Searchers** and **B. Contributors.** Without logged in, user can search for different shows or view all shows. After logged in, she can additionally add a new show, edit an already existing show, or “like” a show.

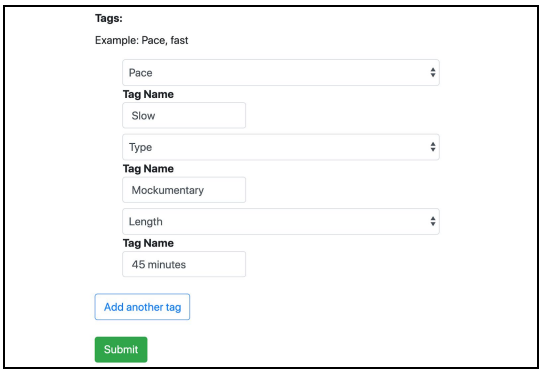
A. For Searchers (Public)

1. Search

When a user enters the website, the home page allows the user to search for the show using different criteria: Title, Creator, Genre, Network, Content Warning and Tags (Pace/Length/Type).

Simply type in or select a search term in the form. As of now, the user can only search for one search criteria, but they can search for multiple tags.



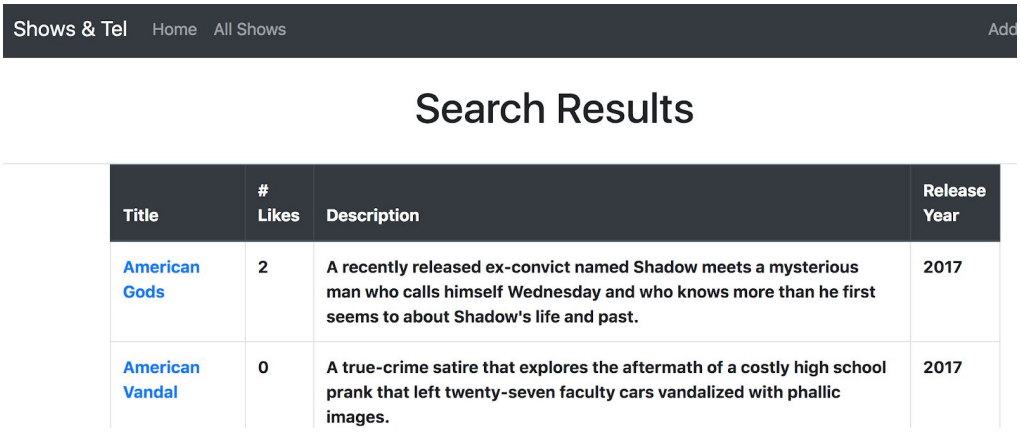


2. View All TV Shows

Without logged in, the user can also view all the shows in the database by clicking on the “All Shows” button on the navigation bar.

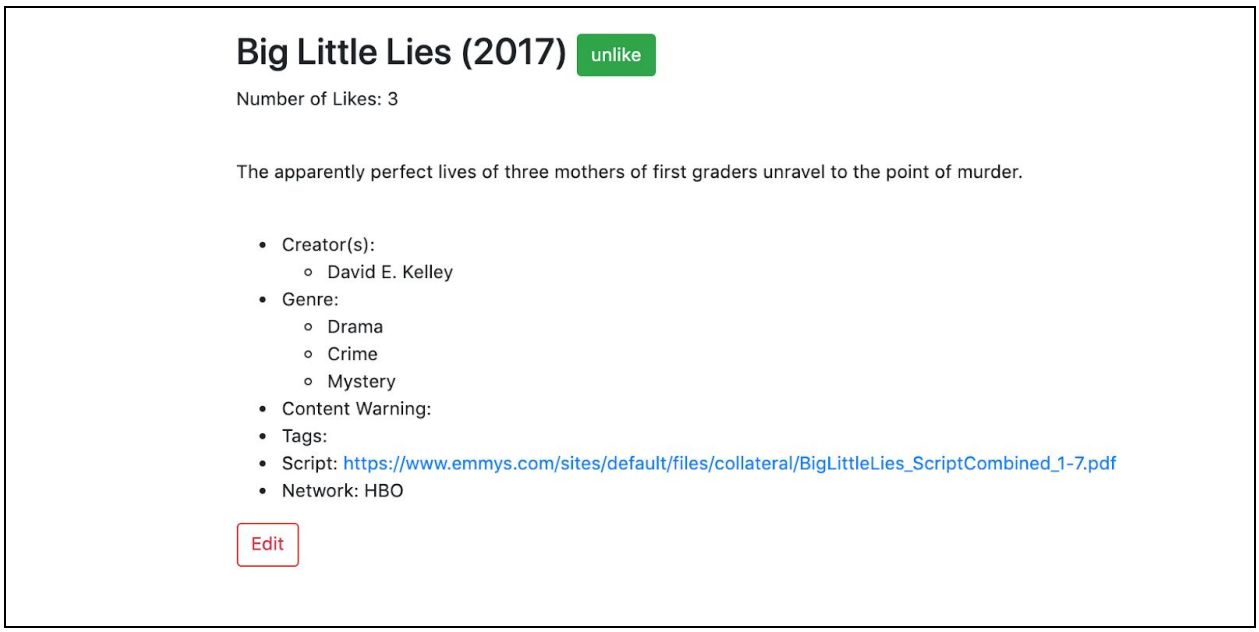
3. Search Results Page

When the user searched for a show or clicked on “All Shows”, the results page displays all the relevant TV shows in a table. Each row displays the name, number of likes, description (logline), and release year of the show.



4. TV Show Profile Page

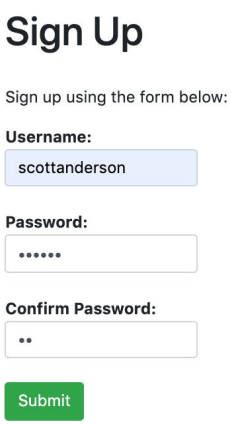
If interested, the user can view some more detailed information about the show by clicking on the name of the show from the Results page, which is linked to the show’s profile page. Each profile includes a title, number of likes, description, creator(s), genre(s), content warnings, gags, (link to or file for) script, and network.

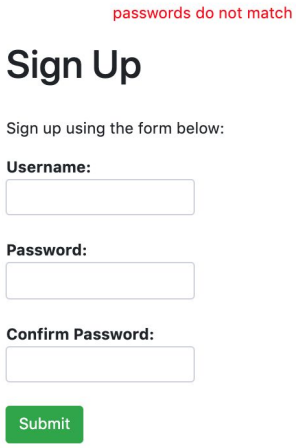


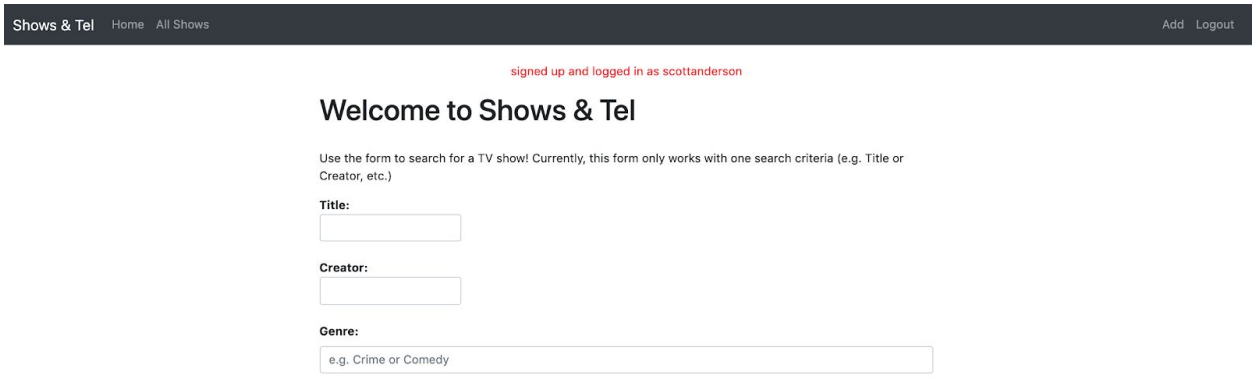
B. Exclusive for Contributors

5. Sign Up & Login to improve our database!

A logged-in user can have more meaningful interactions with our web application and contribute to the class's database. In order to sign up, the user can click on the "Sign Up" button on the right side of the navigation bar. Currently, Shows & Tel. allows anyone to sign up and interact with our web interface. After entering a username and password (twice), user is signed up and automatically logged in, if the username is not taken and passwords match. A message will flash if there is an issue with signup. If successfully signed up and logged in, the user is redirected to the home (search) page, with a flash message indicating her username. Now the buttons on the right side of the navigation bar are changed; instead of "Sign Up" and "Log In", now it is "Add" and "Logout".





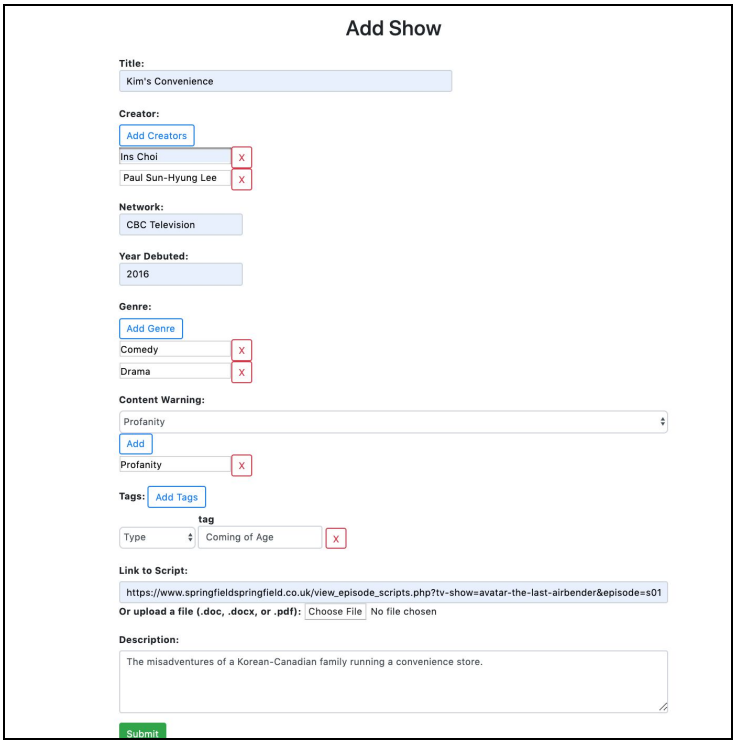


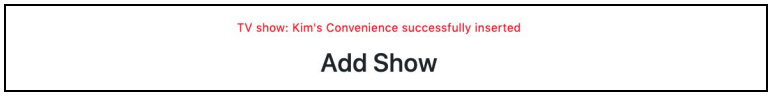
6. Add a TV show

Now that user is logged in, they can add a new show into the database by clicking on the "Add" button on the right side of the navigation bar. The user should fill out all the fields in order to successfully insert a show to the database. Here are the required fields: title, creator, network, year debuted, genre, content warning, tags, link to or file for script, and description.

If user wants to add more than one creators, content warnings, genres, or tags, they can conveniently do so by clicking on the "Add" button next to each field. If user accidentally added a new field and wants to delete it, she can simply click on the "X" button. However, user is

required to put in at least one piece of information per field. Simply hit “Submit”, and flash message will inform whether the show is successfully inserted or not.





7. Edit an existing show

Moreover, a logged-in user can edit and improve TV show profiles. First, the user should navigate to a show profile page either by clicking on the “All Shows” button or searching for a show and clicking on the name of the show. If the user is logged in, they will able to see the “Edit” button on the profile page. Clicking on this button will redirect to a form, which is pre-filled with current information of the show. User can modify them by changing the input fields. If the user wants to add new genres, creators, content warnings, or tags, they can simply do so by clicking on the “Add” buttons next to each input field. After the user modifies the form, simply hit “Update”.

Update Show: Kim's Convenience (#128)

You may delete or add content warnings and creators. However, a show must have at least one content warning and creator.

Note about tags: if you would like to update a show's tags, use the dropdown menu to choose a new tag type and enter a value into the corresponding textbox. However, if you would like to leave a tag unchanged, you need to re-enter the tag type in the dropdown menu. For your reference, here are this show's tags:

type, Coming of Age

Title	Kim's Convenience
Network	CBC Television
Release year	2019
Description	The misadventures of a Korean-Canadian family running a convenience store.
Script	https://www.springfieldspringfield.co.uk/view_episode_scripts.php?tv-show=avatar-the-last- **Or upload a file (.doc, .docx, or .pdf):** Choose File No file chosen
Genre	CS 304 Add Genre
Genre	Comedy X
Content Warning	Profanity Add Warning
Creator	Ins Choi Add Creator
Creator	Catherine Chen X
Creator	Chloe Moon X
Creator	Alice Zhou X
Creator	Scott Anderson X
Tags	Choose a tag type **Tags** Coming of Age Add Tag

update

Kim's Convenience (2019) like

Number of Likes: 0

The misadventures of a Korean-Canadian family running a convenience store.

- Creator(s):
  - Ins Choi
  - Catherine Chen
  - Chloe Moon
  - Alice Zhou
  - Scott Anderson
- Genre:
  - Comedy
  - CS 304
- Content Warning:
  - Profanity
- Tags:
  - : Coming of Age
- Script: https://www.springfieldspringfield.co.uk/view_episode_scripts.php?tv-show=avatar-the-last-airbender&
- Network: CBC Television

Edit

8. Like a show to show some love!
“Like” is another way to make our web application more interesting and interactive. Each user can “Like” or “Unlike” shows, and the “Number of Likes” displayed on the search results page and show profile indicates popularity of the show.

In order to “Like” a show, the user can simply click the green “Like” button on the show profile. Notice the “Number of Likes” described below incremented by 1. If the user decides that the show is actually not so great, they can simply click the same button, which changed to an “Unlike” as soon as the “Like” button had been pressed.

Technical Overview

Shows & Tel. is built on a Flask web application that is written in Python. For the front-end user interactions, we use HTML pages templated through Jinja2, and make use of JavaScript, jQuery, and Ajax.

Tables

The structure for the multiple TV show tables was created using mySQL. Our tables can be categorized into four main categories.

1) **Shows** table
The main “shows” table holds relevant information about the TV show, such as the show id, title, release year, description link to script, and number of likes. It also holds a foreign key of the show’s network from the networks table.

2) Tables for the categories with many-to-one (**networks**) and many-to-many relationships (**creators**, **genres**, **content warnings**, **tags**) with shows
Separate tables were created for categories that exhibit many-to-one or many-to-many relationships with shows. This prevents wasting space and time to record duplicate values when, for example, two shows are both aired on Netflix. Instead of recording “Netflix” every time a show is created by Netflix, the network id is referenced from the networks table. The networks, genres, creators, and contentwarnings table hold an id and the name. For instance, one row of the networks table would hold a network id 1 and name “Netflix”. The **tags** table is structured slightly differently; it holds the tag id, name of the tag (ex. pace, length, type), and the value of the tag (ex. “mocumentary”).

3) **Intermediate tables** for the shows (**showsCreators**, **showsGenres**, **showsWarnings**)
The intermediate tables hold the show id and the id of the relevant field. For instance, the showsCreators table holds the show id from the shows table (foreign key) and the creator id from the creators table (foreign key). Intermediate tables with ids are convenient when recording many-to-many relationships.

4) Other tables: **userpass** and **likes**
The **userpass** contains the user id, username, and password. The **likes** table records the user id and the show id. We insert or delete the record of the relationship whenever a user presses the “Like” or “Unlike” button.

The data for the TV shows was sourced by Wellesley College students by asking them what the shows they watched. This data was then imported through multiple CSV files.

Search Interface & Results Page

The main page of Shows & Tel. displays a form where users can search for TV shows using different criteria such as title, creator, network, tags, etc. The dropdown menus of networks and content warnings were populated from the networks and content warnings table, respectively, through Jinja2 templating. Currently, users can only search for a show using one criteria, with the exception of tags, in which users can search using multiple tags. Submitting the HTML form of search criteria sends the data by POST method to query for the relevant results. Since the query contains several user inputs, these inputs are validated in the Python code to prevent SQL injections. Once the Flask app successfully executes the query, the page redirects to a seperate Results page, which displays the results of the search, in which the entries are populated with results from the SQL query. On the results page, clicking on the title of a show redirects the user to a show’s profile page.

Adding a New Show

When a user clicks on the ‘Add’ button in the navbar, they are redirected to a page that houses an empty form where they can add a new show into our database. Some cool features on this page are the dynamic add buttons for creators, genres, content warnings, and tags that are similar to the ones on the search page. When clicked, these buttons invoke JavaScript functions that dynamically add HTML elements to the page so the user can input more than one value. The form requires that all information boxes be filled out, with tags being the only optional input. When the ‘submit’ button is clicked, the inputs are validated for existing inputs and on a successful check, the values are entered in our database tables. If there are any incomplete fields, the edit page will reload with an error message. On a successful submit, the page will redirect to the new show’s profile page.

Script Upload/Links

In both the ‘Add’ and ‘Edit’ pages, the user has the option to upload a file for a TV show script if they do not wish to include a URL link. If a user provides both a URL and a file, then the default script that is used is the provided file. The user can upload files of type .doc, .docx, and .pdf. There is currently no file size limit, but if we had time we would add validation in both the frontend and the backend, in the form of a directional message and file size check in Python, respectively. Cloud9 only allows files up to 8MB, however, TV show scripts have the potential to be much larger sizes. One potential fix is to store these documents in a different place (currently we are storing them in the Cloud9 filesystem), for example as blobs in the database, or use AWS to store them in the cloud. On a successful file upload, the file is stored in an uploads folder in our filesystems and the filename is added to the shows table for its respective show.

When a user visits a show’s profile page and clicks on the script link, the following action depends on what type of script is stored in our database (either a URL or a file). On a script link click, it invokes a script() function which determines where the location of the script is. If a script is local, aka stored in our filesystem, then we render it the normal way by passing the filepath to our profile template. If the script is external, aka we stored a http link in our database, then we do a straight redirect to that stored URL.

Editing a TV Show

When a user decides to edit a TV show by clicking on the ‘Edit’ button on a show’s profile page, they are redirected to a new page that displays a form that is rendered with the existing show information (with the exception of tag type). This is done with a SQL query. The edit page is similar to the add page; the frontend components are essentially made with the same HTML/JavaScript skeleton/functionality, but the way the information is handled in the backend is different. Instead of inserting all the information on the edit page, our application determines which values are new and after validation, updates the database tables with the new information. When updating warnings, the backend function compares the list of old warnings obtained from the database and new warnings submitted by the form. Comparing the old and new warnings yield two new lists, a list of warnings to add and to delete from the intermediate tables are created, and they are added and inserted, respectively, with a SQL query. Genres and creators are updated with the same process. On a successful update, the page redirects the user back to the show’s profile page, which displays the updated information.

One note about the script link/upload option is that if the script information is updated by either a new link or file upload, on the redirect after a successful update, the new script may not appear right away due to browser caching. The browser will use the old link reference until the reference in the cache expires. If a user would like to immediately view the new script, the user must hit SHIFT-REFRESH in order to refresh the cache.

Likes

The likes feature makes use of AJAX and JQuery. Clicking the like button triggers a callback function that sends information such as username of the person, the show they are liking, and whether the show was liked or unliked by the user. There is a mySQL table (“likes”) that references that information, and in accordance, helps check whether the user is logged in and gives the info for the show that they are liking. Once the Flask app gets the information, it executes the SQL queries to insert the show id and user id into the “likes” table. Additional SQL queries compute the number of likes of the show and update the shows table. Once the queries are executed, it returns a JSON response to the request, and the callback function updates the number of likes of the show. If the user is in session, the likes for each show can be updated and incremented/decremented. Each user can only like a show once.

Signup & Login/Signout

The signup and login features make use of a mySQL table that stores a username and a hashed password. Signing up creates a POST request, and the code checks if the passwords match. If they do, the password is hashed. It also checks if the username is taken in the table, and if it isn’t, the new username is added to the table. The session is now marked as in session for the username. The login form works similarly, but instead of adding information to the table, it checks if the username and password are correct. Finally, the logout form stops (pops) the session, and the user is formally logged out. Users are unable to like, edit, and add shows if they are not logged in. However, they can still see all the TV shows that are currently in the database and their profiles.

Conclusion

In conclusion, our database serves the Wellesley community by helping the Wellesley screenwriting and TV show classes. The features such as the tags, content warnings, and uploading the script make for a website/database that assists writers with finding and adding scripts with relative ease. In addition, it is scalable in the future for a wider audience and aspiring screenwriters everywhere, as it is an incredibly helpful tool to have for writing one’s own screenplay, and that is not limited to Wellesley.

Note: If time permitted, one feature we wish we could have implemented is multiple search criteria. We would have done so by using string concatenation to query the database. For every non-empty input from the form, we would have added more clause in our search query. There would mainly be three different parts. First, we would have a starting query, such as “select * from shows where “. Then for each user input, we would concatenate a string, such as “nid = 3” (nid refers to network id) and “year = 1996’. However, because we have 6 different search

criteria, 3 of which have separate tables and intermediate tables, we would have to consider all different cases and search for the intermediate tables accordingly. Or we could have constructed a VIEW table or join all the relevant tables together and search from that table.

Additionally, we wish we could have enabled users to add a new input even if the field is empty. Because we obtained real data from the students, there are many shows without content warnings and tags. Currently, a user is unable to add a new field if the show doesn't contain that field (ex. warning, tag) at all. We would have checked for an empty value in the edit template and, if so, populated a row with an empty textbox.