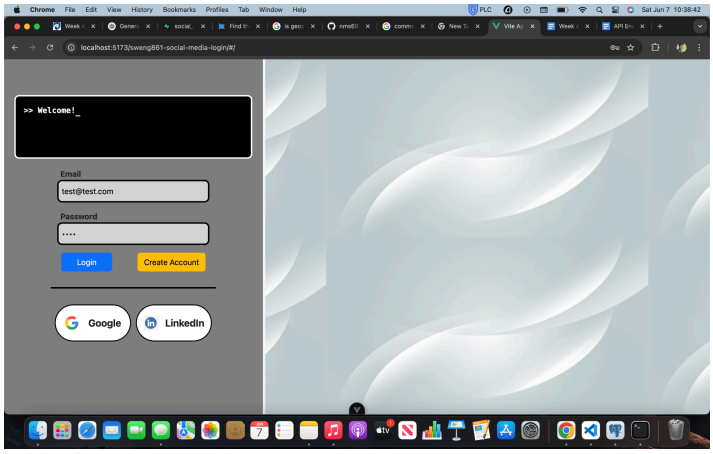
Introduction

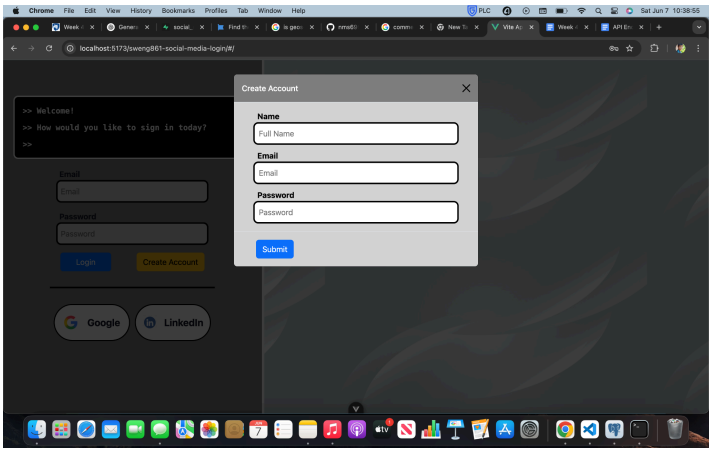
This document outlines the user interface and how it interacts with the backend API endpoints to perform CRUD operations.

Implementation

Login + Account Creation

When the user first visits the application, the user is presented with three options to login. They can login with their google account, their linkedin account, or they can choose to login with a manual set of credentials maintained from the application itself. Attached below is a screenshot of the options.

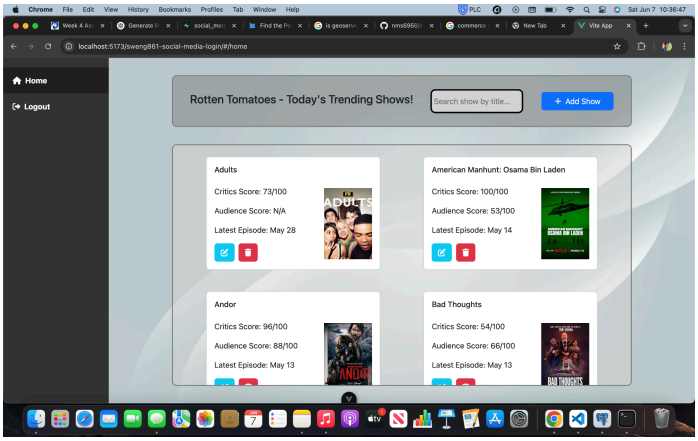


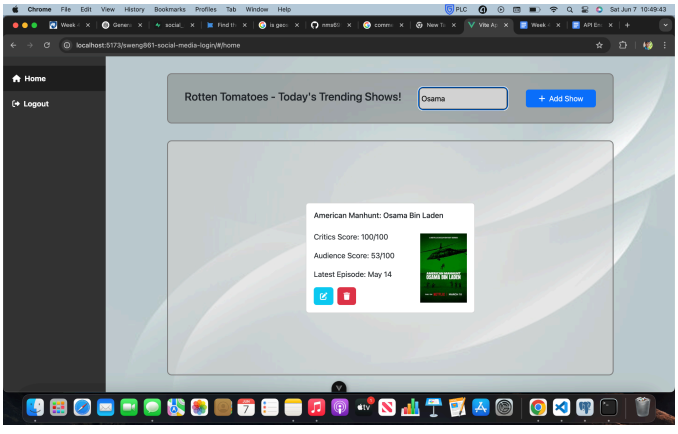


If the user visits the page for the first time, and thus does not have a set of established credentials yet, they can choose to hit the “Create Account” button. This will trigger a form that pops up where the user can fill out their name, email, and password. See the screenshot above of the Create Account component. When the user creates an account, the user's data is inserted into the database. Part of this inserted data is the hashed password. When the user then proceeds to login with the account they just created, the password they input on login is hashed and compared with the stored hashed password in the database. This ensures that no raw passwords / private data is stored in a database anywhere.

Home Screen + CRUD Operations

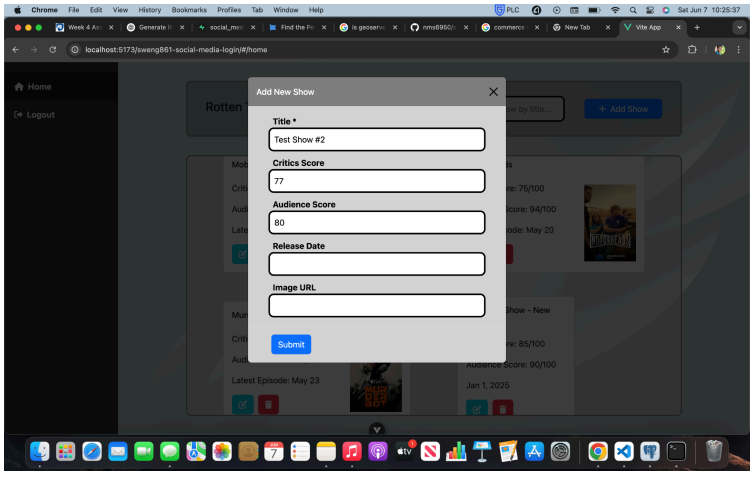
After the user logs in to the application, independent of which login service they prefer, the user is redirected to the /home route. On this page, they will see a dashboard with all the tv shows saved in our tv_shows database. See the screenshot below.

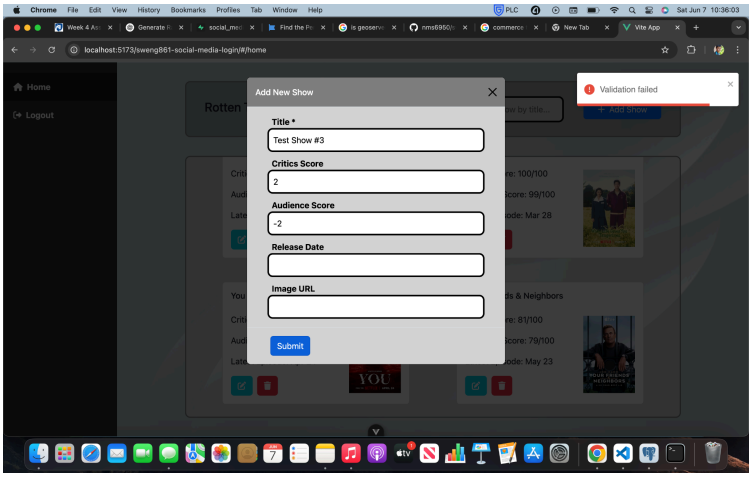




This home screen contains a header at the top describing the purpose of the application, which is to show the shows that are trending lately on Rotten Tomatoes! Additionally in the header, there are the options to search for a show by title, and add a new show button. The search bar works in a way such that whatever the user inputs, the dashboard filters the shows based on whether or not the inputted string is a substring of any show title. See the screenshot above for an example of how the search bar works.

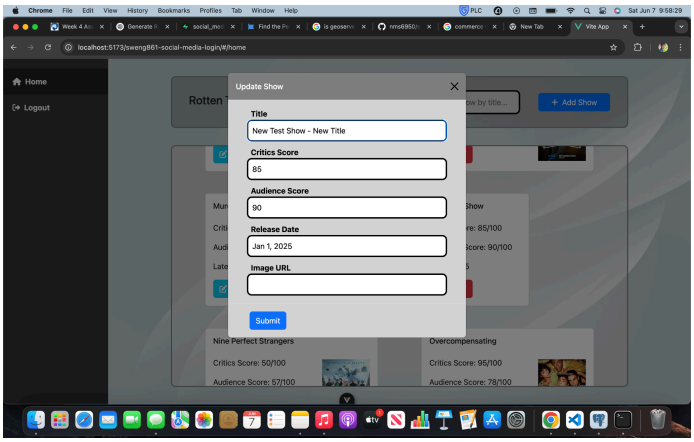
When the user hits the “Add New Show” button, they are presented with a form to fill out. The form contains all the necessary information to create a new entry in the database: title, critics score, audience score, release date, and image url. Not all of this information is required to create an entry however, actually only the title is needed. As listed in the API Endpoint Documentation, there is a data validation check on creating a new show for the correct data variable types, values, etc. See the screenshots below of the form and error validation.

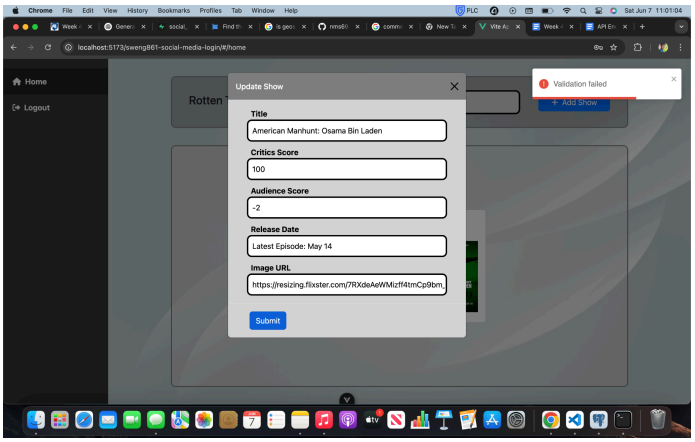




If the user fills out all the correct data, and it passes validation, the entry is inserted into the database. The user is then brought back to the home screen, and the shows displayed are refreshed, and the user will be able to find the new entry they just made.

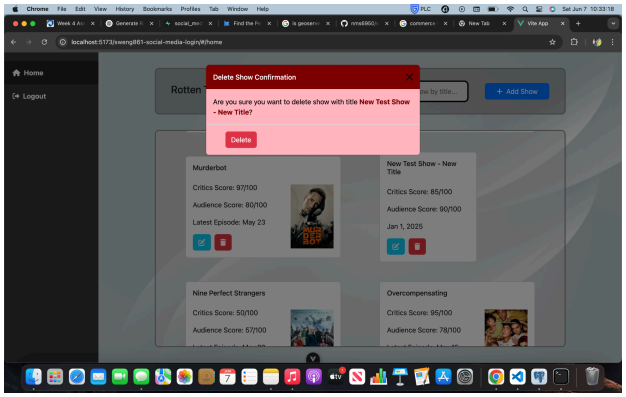
Within each show card, there are two buttons in the bottom left hand corner. One button is a pencil icon, indicating the user can update a show entry. The other button is a trash icon, indicating the user can delete that show entry. When the user clicks on the pencil icon, an update show form will appear. This form contains the same fields that the create new show form contains. Similarly, this form works the exact same way as the add new show form, except it contains different data validation checks, as listed out in the /updateShow endpoint in the API Endpoint Documentation. If the user clicks on the update form, fills out an invalid data value, and clicks submit, the application will throw an error similar to the new show errors.

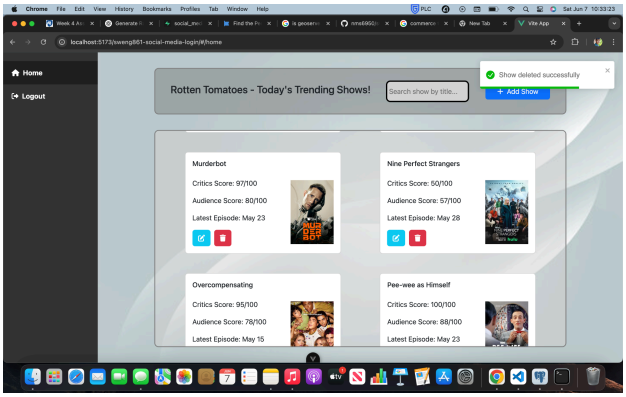




If the user opens the update form, updates any value with valid data values, and clicks submit, the application makes a request to the /updateShow API endpoint, and updates the entry for that show. After a successful update operation, the application returns the user to the dashboard, refreshes the displayed shows, and the user can see the show they updated with the updated values.

When the user clicks on the delete icon, the user is presented with a confirmation message. I decided to implement a confirmation message functionality here to prevent any user errors that can occur if they accidentally hit the trash icon when they didn't mean to. The confirmation message ensures that the user actually wants to delete the entry and requires them to execute two distinct submit events to do so. If the user does want to actually delete the show, they will hit the trash icon, be presented with the confirmation popup, and then hit the delete button. See the screenshot below of the confirmation pop-up. When the user goes through the delete process, the application makes a delete request to the /deleteShow API endpoint, and the show is deleted. See the screenshot below of the success message after deletion.





Navigation Bar

Last part of the user interface, as depicted in multiple screenshots above, is the navigation bar located on the left hand side of the screen. As of now, the navigation bar only has two entries… home and logout. When the user is on the home screen and hits the home link in the navigation bar, the application essentially does nothing because it redirects the user the page they are already on. If the user hits the logout application, the user is logged out, and they will be redirected to the login page.