Project summary

We want to provide our users with the opportunity to search through a list of products/items/names, choose their favorites, and have them saved off for later use. This should be built with responsive design in mind and work functionally across all major browsers (Chrome/Firefox/Safari/Edge).

Project requirements

Part 1 – API Integration (Required)

- Create a search component that utilizes the API of your choice as its backend. Example APIs could be Marvel or the Lord of the Rings API but use whatever works best with the time allotted and is of interest to you.
- The user should be able to enter a search term and have results filtered automatically without having to hit a search button or press enter.
    - When no search term has been provided, provide an unfiltered list of results
    - We do not require pagination currently and if the API returns more than 100 results, limit your display to 100 results.
- Your backend code should consume your chosen external API, and your frontend code should call your custom backend endpoint
- The results should be displayed in some form of grid below the search area
- Each item should include, at minimum, the name of the item.
    - If the API also contains images, include the image with the name, if it does not, omit them
    - Feel free to include any additional information you find relevant for the user to select between items
- When a new search term is provided, update the list of results
    - This should be handled in JavaScript without reloading the page
    - It should use the same grid style and name/image requirements as the initial page load

Part 2 – Favorites (Extra Credit)

- The user should be able to create multiple lists to save search results.
    - If the chosen name already exists, an error message should be shown prompting the user to select a new name
- The user should be able to 'favorite' any number of the search results from above and save them to one, or more, of their lists
- Provide a view of these lists.
    - List should be displayed in alphabetical order
    - Clicking on a list should display the favorites

    - Ability to delete a list should also be provided in this view
- When viewing a list, favorites should be sorted by modified date
    - Allow for a favorite to be deleted in this view
    - Show the total number of items stored in each favorite list
- Data should be persisted in the storage engine of your choice (MySQL, SQLite, Redis, file, etc…) via calls to an API endpoint
- All page actions should be handled in JavaScript without reloading the page
- An existing list can be deleted from the view
- Testing should be a part of your final submission
    - There is no code coverage goal for this, you do not need 100% code coverage for both front and back-end code
    - Testing, however, is expected and tests that you do write, should pass.

Technology choices

- All front-end JavaScript elements should be developed in a modern JavaScript framework of your choice (Vue, Angular, React, vanilla JS, etc…) and compiled using a front-end compiler of your choice (gulp, webpack, grunt, etc..). Use something you're comfortable with but be prepared to defend your choices.
- All back-end elements can be developed in a modern PHP framework of your choice (Symfony, Laravel, Yii, CakePHP, etc…). Be prepared to defend your choices with this as well.

Expectations

- This project should only take a few hours to complete. If you find yourself spending more than that, finalize what you do have and outline a plan for how you would finish the project and how much time you expect it will take to finish.
- You ARE NOT expected to build out an entire webpage for this. EG: you don't need fully formed navigation, footer, etc... However, your components should be properly styled and visually pleasing.
- You DO NOT need to build out any user authentication/login for this. You can safely assume that these components will be pulled into an application that already has authentication defined and that those checks will be handled higher up in the application.
- If you have docker experience, preference is to build this out with docker, however, this is not a requirement, nor does it have any impact on how the project will be evaluated should you choose not to build it using docker or do not have experience using docker.
- Implementation will be graded on execution, organization of code, maintainability, optimization, scalability, security, tests passing, and readability.

Submission

Submit your work as a Git repository with instructions on how to build your implementation.