Dev Review:

Branches:

Original scratch notes:

- Everything is on develop branch and not on main
- CICD pipeline on develop or main???
    - Created two yml files for master and develop
    - Will need to make sure to delete the develop after it's been merged

Organized notes:

Working on a separate branch is fine, but there are some things you will need to consider.

1. Make sure when you merge to main/master, there will be no conflicts. It's gonig to be really hard to fix those merge conflicts if since there were huge changes
2. I noticed the CICD pipeline inside .github has two yml files. 1 for Dev and 1 for master
    a. This is fine, but I think a bit unnecessary. What could be done is to have one yml file and have:
        i. on:
        pull-request:
        branches:
        [main, develop]
    b. Either way you will have to delete your develop yml, unless you're keeping that branch until the end of the project
    c. I'd recommend cleaning up the repo when you're done ie delete all those unnessary branches

Deployment and firebase use:

Original scratch notes:

- Using firebase for deployment
    - Used because they are familiar

Organized notes:

I recognize that using firebase was helpful for deployment, but I think it's unnecessary if you are ***only*** using it for deployment. One benefit is that you are allowing for potential expandabliilty since a future might be able to just keep using it for adding a backend. In fact, if it's only used for deployment, then it really doesn't make it that much more expandable since the actual database isn't set up.

I don't believe that this app necessarily requires having a database since that could be overkill, but there are many other very simple deployment services that you could use.

Since you are already working with local storage, it seems that you could have added a database and created a REST API for it.

I have spoken to the professor about using a database, and he does agree that it can be overkill for many apps, so in this case, it's definitely fine.

TL;DR: using firebase is fine, but can be misleading since it's primarily a backend SDK. For example, firebase offers authentication, RealTime database, etc.

Code:

Original scratch notes:

- Refactored all the scripts to be called in one main.js
- objName in localStorage.js
- deleteFromLocal(key) function:
    - Why are we deleting obj and then saving the obj into storage?
- Created a function called inputSanitizer()
    - This is used once and only checks for an empty input, why create an entire function for this?
- Where is local storage and the front end to save tasks connected?
    - Yes it is inside another branch

Organized notes:

I liked that you refactored all the code into one single main.js. It definitely makes it a lot cleaner.

I think the naming inside /source/modules/localStorage.js could be a bit more clear. Example:
objName, storeToLocal(**key**, **value**), obj, etc.

objName seems to be the task name, so why not call it taskName? I recognize that local storage might generally have a convention to use obj as names, but I'm not too sure. If using obj is a very common convention for local storage, then leave it as is.

The parameter names: key, and value could be a bit more clear, but they still make sense overall; however, this does bring up a question now… What is key? What is value? Is key the task name? Task id? Is value task description? What exactly is being stored into local storage?

Does it look like this? Example:

```
{
        "task": "task description"
}
```

Local storage only saves key-value pairs, so it does make sense to call it that, so you can probably keep it the way it is, but I think it could be helpful to understand what the key-value pair is.

There are two functions inside local storage that I don't really understand, but that doesn't mean its wrong by any means. deleteFromLocal vs removeDataFromStorage

In deleteFromLocal, you are deleting the key and in removeDataFromStorage, you are deleting the actual object.

What is the purpose of deleting the key? Are you using that to overwrite it? It seems that you delete the original key, and now overwrite it with a new key-value using saveToStorage? If you are overwriting something, should it be called updateLocal? Not too sure because I don't really know what's being saved into local storage as the key and value. My assumption would be that the key is the task id or something and the value is the actual description of the task.

TL;DR: if deleteFromLocal is overwriting an existing task, then i think it should be called update or something. If it's used for a different purpose and it's accurate to it, then leave it as it is.

```
/**
 * @name deleteTaskFromLocal
 * @function
 * @description remove a key from local storage
 * @param {string} key task name
 */
function deleteFromLocal(key) {

    const obj = retrieveDataFromStorage(objName);
    delete obj[key];
    saveToStorage(objName, obj);
}
```

```
/**
 * @name removeDataFromStorage
 * @function
 * @description remove the object from local storage
 */
function removeDataFromStorage() {
    localStorage.removeItem(objName);
}
```

I saw that there is an inputSanitizer function inside task-list.js. I don't see why this is it's own separate function for a simple check. All it's checking is if there is input: if no input return true, other wise return false. If you plan to use inputSanitizer more, then it could be helpful, but in my opinion, it's unnecessary. I recognize that you want to take advantage of modularization, but since this is such a simple function, it actually seems to take more time to investigate what that is rather than simply just doing:

if (!taskName.input) {...code…}

Having inputSanitizer actually made it more confusing since I thought you were cleaning the input.

Overall, inputSanitizer isn't a bad idea if you plan to reuse it and actually do a bit more with it, but if you only plan to use it for checking for empty input, it's unnecessary.

I also had an original note about connecting the local storage to the front end, which has been clarified on another branch for task-list.

Inside skins-themes.js, don't forget to document it!

I noticed that in the app, you are pulling a lot of data from the repo/github: music, background themes, etc. I had made the assumption that since this is happening, the actual site will be slow, and when I went ahead and checked out the site, I noticed it was a little slow. I'm not entirely sure how you can get around this, but I think using firebase database could have potentially helped with this. The music files are roughly 3 mb - 12 mb which might be causing the site to slow down. I'm not really sure if using a database would fix this, but maybe it could have. Is there a way to shrink the file or encode it to be smaller? I don't necessarily mean to shrink it by compressing it, instead, is there a way to encode it into some kinda of binary file or something? I'm not super familiar with this

concept, but I know that there are ways to encode jpeg images to be binary and it makes it a lot quicker.

Testing:

Will you be doing E2E testing or just unit tests? I'd suggest speaking to your TA and figuring out which would be the higher priority.

Overall:

This is a really good app. I think things are looking pretty code and you guys are on a good track. Many of my feedback is just clarification questions for you to think about. Nothing that I see is "wrong," there are just questions about the thought process and reasoning.