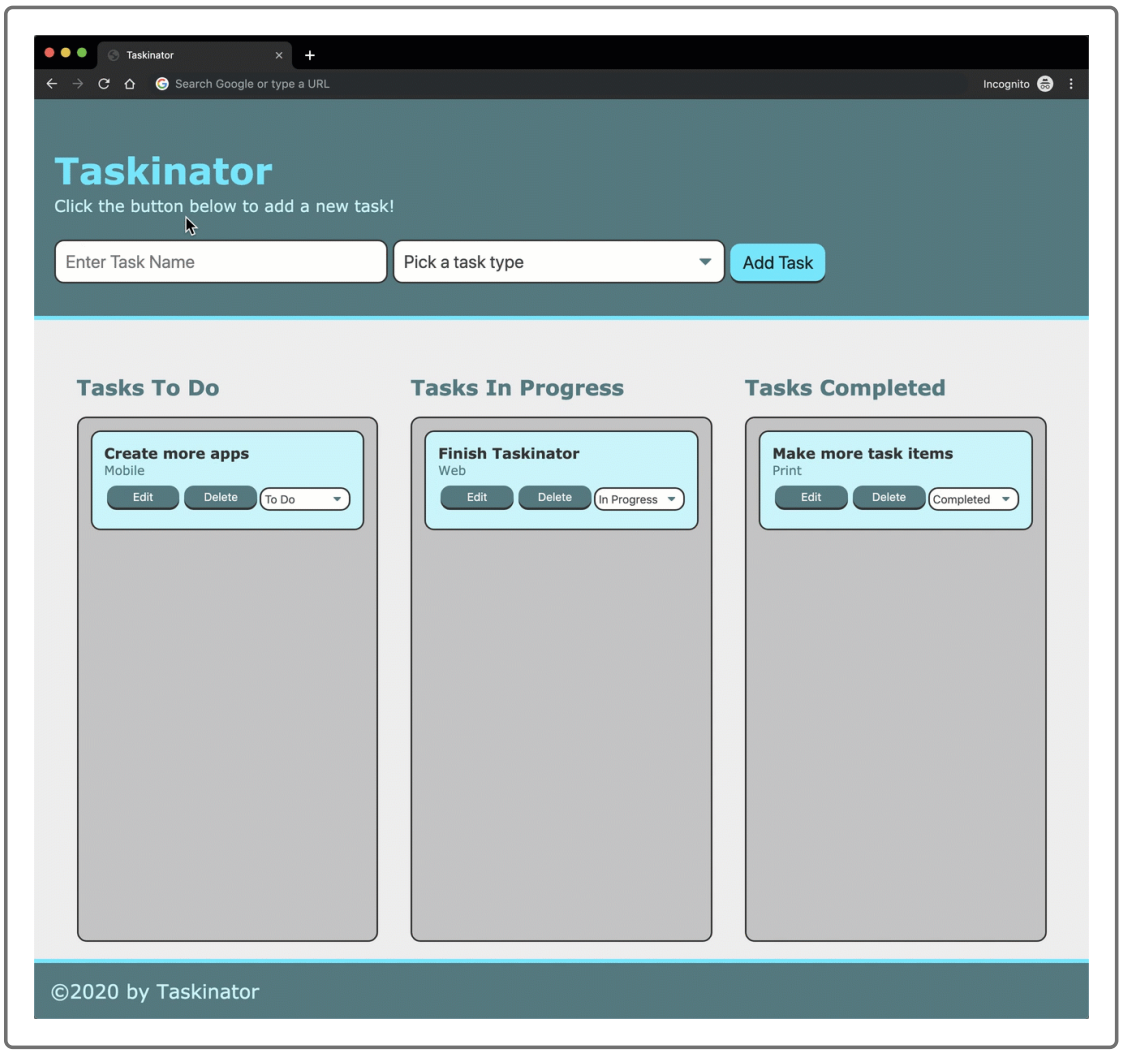
4.5.2 Preview

Adding persistence with `localStorage` will enable us to keep tasks in their corresponding status lists after refreshing the page. See this animation for a demonstration:



The UI and core functionality won't change, but this new feature will make Taskinator more usable. However, we'll need to restructure the app's data to make it easier to save and retrieve from `localStorage`.

Let's consider the best approach to accomplishing this goal:

1. Create a new Git branch. We'll create a feature branch that corresponds to the GitHub issue we're addressing in this lesson.

2. Save tasks to an array. We'll organize all the elements that correspond to a task into an object, and save those objects into an array.

3. Save tasks to `localStorage`. We'll implement the ability to save our tasks to `localStorage` so that they can be retrieved later.

4. Load tasks from `localStorage`. We'll add the ability to retrieve saved tasks from `localStorage`.

5. Optimize the code. Our app is working as we want it to now, but we'll take the time to go back and lessen our technical debt by refactoring the code. This is a best practice that it's good to get into the habit of!

6. Save our progress with Git. Save our progress with Git. We'll merge the `feature` branch into the `develop` branch and push our changes to GitHub.

7. Deploy to GitHub Pages. We'll deploy our finished application to GitHub Pages, so we can show it off to the world and use it anywhere!

In the first part of the lesson, we'll set up the `localStorage` functionality. During that process, we'll create similar functionality to code we've already written—leaving the application with some technical debt. We could leave the code as is, but this will be a good opportunity to refactor and reuse the already existing functionality.

Let's get started by creating a feature branch for the issue.