step 1:

// DOM event listner : Load all previously existing DOM entries

// DOMContent Loaded is an event that gets loaded after the DOM is created i.e. when the page gets loaded

step 2:

// Add Task event

// Store in local Storage too

step3:

// Remove task event - each specific task by clicking on x icon

// and remove it from local storage too

step 4:

//Remove the entire list by clicking on clear Tasks Button

// and remove it from local storage too

step 5:

//Filter the Task events

step 6:

/\*Additional note: the arguement 'taskInput.value' has been replaced by task because there is NO input value coming from user BUT only the task info which is already stored in LS' \*/

step 7:

/\* removeTask: remove task Since they are multiple items and can be dynamic in nature if.e. they can be added as per user discretion, we'll havr to rely on event delegation and we need to use the event listener on parent UL with the class "taskList" itself \*/

step 8:

/\* This is being removed from the DOM through the above code.

parent of the parent is the LI item which needs to be wiped off when the click happens \*/

step 9:

/\* Remove from LS(Local storage) \*/

/\* there is no particular ID that can be passed as an argument for easy hooking, thus treating the very li that has been created above as the refrence point \*/

step 10:

// taskList.innerHTML = "" /\* can do this or below - both work \*/

// but the while loop is faster: per the defination below, the loop would continue to work till there is s a first child of the taskList

step 11:

/\*In below, you can use forEach because querySelectorAll returns a NodeList and not HTML collection and forEach() can be used on NodeList. The same would NOT be true for getElementsByClass or tag because that wouldve returned HTML collection and then you wouldve had to convert it to array first in order to apply forEach() \*/