# Javascript Assignment

## Units

**ICTWEB411 - Produce basic client-side script for dynamic web pages**

## Scenario

A client requires the development of a simple application to assign a list of items into two different categories, perishable and non-perishable. The items will then be displayed under two different lists named after the categories. The sorted lists need to be made persistent, so that the data stays in the same state when the application is reopened / restarted.

Requirements:

1. The master list of tasks need to be displayed. The data for this will be supplied.
2. The categories need to be shown with items in the category
3. Items that have been assigned a category needs to be removed from the master list
4. An item can be removed from a category and it should reappear in the master list so it can be reassigned
5. Under each category the list needs to be organised alphabetically, as well as in the master list
6. The application needs to store all data so the sorted lists can be persistent (can stay the same after page is reloaded)

## Task

### Summary

1. Analyse requirements
   1. Create a wireframe showing the proposed interface suitable for the task
   2. Create a pseudocode sequence to show the working of your application, outlining the task the user needs to perform to sort the items.
2. Create project and assets
   1. Create a layout in HTML and CSS to reflect your wireframe design
   2. Create a script to facilitate the required functionality
3. Test and debug report
   1. Create a table of issues you encountered while working on your script or your HTML and CSS document. See the Test Report section

## Test report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Functions | Description | Test Case Total | Test Case Passed | Remarks (explain what happened) |
| Loading data | loading data from file |  |  |  |
| Displaying data | displaying data in the master list interface |  |  |  |
| Add item to Perishable list | adding an item to the perishable list |  |  |  |
| Add item to Non Perishable list | adding an item to the non-perishable list |  |  |  |
| Item removed from master list | item being removed from master list after being added to perishable list |  |  |  |
| Item removed from master list | item being removed from master list after being added to non-perishable list |  |  |  |
| Removing item from perishable list | item being added to master list after being removed from perishable list |  |  |  |
| Removing item from non-perishable list | item being added to master list after being removed from perishable list |  |  |  |
| Data persistence | application maintaining state after the page being reloaded |  |  |  |

Use the above table to test your application.

1. Each item needs to be tested **at least 5 times** and result recorded.
2. If the test case passes, record under Test Case Passed
3. Remark about why it failed and how you might fix it. Eg “...not loading data, need to check data file and path to file in the HTML...”

## Submission

Submit a report to Moodle containing

1. Your **wireframes**
2. Your **pseudocode**
3. A link to your github repository, which contains your:
   1. HTML file
   2. CSS
   3. Javascript
4. Your test results

### Due date

Monday 04/05/2020

## Marking Table

### Overall result

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Description | Weight | Result |
| Analyse Requirements (wireframe) | Create a wireframe for the interface | 15% |  |
| Analyse Requirements (flowchart) | Create a pseudocode of the required algorithm | 15% |  |
| Create project and assets | Create a working prototype of the application | 55% |  |
| Test report | Fill out test report | 15% |  |
| TOTAL |  | 100% |  |

## Performance Criteria

### Analyse the requirements for web documents

* 1. Determine the necessary dynamic functionality of the web document
  2. Determine the appropriate language to achieve that functionality
  3. Determine the web document requirements

### Design and produce web documents

* 1. Design the web document, and embedded scripts to achieve the required functionality
  2. Write a simple hypertext markup language (HTML), considering accessibility
  3. Write embedded scripts

### Test the scripts and debug

* 1. Test the web document against the required functionality, and reiterate until correct
  2. Complete the documentation and submit it to the appropriate person for their approval

## Unit Mapping

|  |  |
| --- | --- |
| Task | Criteria |
| 1 | 1.1, 1.2, 1.3 |
| 2 | 2.1, 2.2, 2.3 |
| 3 | 3.1, 3.2 |