TodoMVC Test Plan

Version	Author	Date
1.0	Kate & Sean	05/02/2024
0.9 (this version)	T.Tester	10/12/2022
0.8	T.Tester	10/10/2022

Objective

This is a test plan for the TodoMVC browser, to check for high priority bugs that are to be addressed and fixed as soon as possible and any medium or low priority bugs which are to be logged and fixed when time is available.

TodoMVC is a website which collates examples of the same "To-Do List" application implemented in all of the different JavaScript frameworks, for developers to use as examples when evaluating what framework they would like to use for their own application development.

The simple to-do list application doesn't have any persistent storage; it stores its data in the user's local storage, the application is not designed to be a real-world to-do list manager, as there's no way to share the data between sessions, browsers or machines.

Scope of Testing

In:

- The 9 frameworks labelled “new” should be tested against coverage criteria first:
- React, React Redux, Vue.js, React, Backbone.js, Angular, Ember.js, Lit and from compile-to-JS Svelte, as these are all new or updated, so most likely to have bugs.
- We will test using the listed browsers (see below)
- If there is any time remaining, we will begin to look into the other 33 frameworks and the additional 4 non-framework implementations, as they are also new or updated
- Test using desktop browser

Out:

- We will not be testing the “Download”, “View of GitHub”, “Blog” and Twitter(X) buttons
- We also won’t be testing any other links from the page that are not part of the examples list/specified in out in scope section
- We will not be testing on mobile devices
- Keyboard navigation

Coverage Criteria

All of the functional requirements below need to have been tested in the frameworks in scope, for us to achieve 100% coverage:

1. New todo items can be added
2. Todo items can always be modified
3. Todo items can always be individually deleted
4. A todo item can be marked completed or incomplete
5. All todo items can be marked as completed, whether they have been completed or not
6. If all todo items are marked as completed, they can all be marked as incomplete in one go
7. The list can be filtered on todo items' completion states
8. Complete todo items can be cleared from the list, when at least 1 completed todo items are listed
9. Status bar always displays a count of remaining todo items left to do
10. Todo items can be reordered

Test Environments

We will be running tests against the production website (https://todomvc.com) which is unlikely to present any risk as it is still in development and therefor shouldn't have many users.
Tests should be performed in the following browsers:

- Latest version of Chrome
- Latest version of Safari
- Latest version of Firefox
- Latest version of Edge

Test Team

We have 7 testers in our test team, however 4 of them are currently allocated to other projects which are due for release in the coming months. This leaves us with 3 team members for TodoMVC testing. We can discuss using some of the testers that are on other projects if we find there are multiple issues and the workload is too much as the release date is sooner than the other projects.
The testers assigned to this project are:

- A.Adams - Test Team Lead, the most experienced member of the team.
- B.Barker - Test Engineer who's been the primary tester on TodoMVC for the entire project.
- C.Chapel - Test Engineer who joined the company last week and is primarily still completing onboarding.
- If needed D.Duncan - Test Engineer can come and help with workload but we will assess the situation as it develops.

Features To Be Tested

For each of the popular frameworks, we want to verify the following features:

- Add a new ToDo item
  - Can't add an item with an empty value
  - Can add a value with a single character
  - Check that at least 1 type of accented character and special characters are supported e.g. ä, @
  - Check that an emoji character can be supported e.g ☠
- Modify a ToDo item (by double-clicking)
  - If you modify a ToDo item and click Escape during edit, it should cancel the modification
- A ToDo item can be deleted, when mouse is hovered over and X should appear to delete item
- A ToDo item can be marked as completed (it will appear with a line through it)
  - A completed ToDo item can be unticked again
  - Several ToDo items can be marked as complete whether they are completed or not
  - If all ToDo items are marked as complete they can all be marked as incomplete in one go
- A ToDo item can be reordered by dragging it up or down in the list
- Check status bar:
  - Status bar is hidden when there are no ToDo items in the system
  - Status bar displays "0 items left" when there are no items left
  - Status bar displays "1 item left" when there is 1 item left
  - Status bar displays "2 items left" when there are 2 items left
- Status bar can filter ToDo items: toggle between Active, All and Completed
- When there are any completed items, a "Clear completed" link appears in the status bar
  - When the "Clear completed" link is clicked, all completed items are deleted
- Clicking the down arrow symbol next to the "What needs to be done?" box will toggle all items to Completed or Not Completed

As well as testing within each individual to-do list application, we want to test the behaviour across the different variants:

- ToDo items from one variant do not "bleed over" into other applications (e.g if you create a ToDo item in the React version, it is *not* visible in the Vue.js version)
- The master list of ToDo items is stored in the browser's local storage, with the key name "todos-<version>" (e.g. todos-react for the React version). This key contains a comma-separated list of all the UUIDs for the ToDo items.
- Each individual ToDo item has its own entry in local storage, for

instance if todos-react says that there is a ToDo item with UUID b3a592f9-fbfb-6461-4eef-fba1274b5868 then there will be a corresponding key todos-react-b3a592f9-fbfb-6461-4eef-fba1274b5868 containing a JSON representation of that ToDo item.

An example of the desired JSON data for an individual ToDo item can be found below:

```
{
  "title": "Buy some milk",
  "priority": 3,
  "completed": false,
  "id": "b3a592f9-fbfb-6461-4eef-fba1274b5868"
}
```

The properties are defined as follows:

- **title** *(String)* - The text of the ToDo item.
- **priority** *(Int)* - The order of the item in the ToDo list (starts at 1).
- **completed** *(Boolean)* - Whether the ToDo item has been ticked off.
- **id** *(string)* - The UUID of the ToDo item.

Test Estimation

As a team, we have estimated how long it will take to complete a test of the "Features To Be Tested" for a single variant, in a single browser (e.g. just the React version, in Chrome). The team conservatively estimates that it will take around *1 hour* to complete a single test cycle, however this could taker longer if issues discovered. Additional time would then be needed to investigate/log any defects which are encountered along the way.
Therefore we can estimate the amount of time to execute the entire set of tests without bug reporting:

- 9 different “new” variants listed on the TodoMVC homepage
- 4 different browsers to test for each (Chrome, Safari, Firefox, Edge)
- = 36 hours of testing

Dividing this effort between the 3 testers that we have available, with the possibility of a 4th tester if needed, we therefore estimate that it will take us 12 working hours per person split between 2 days, to complete this testing. However, some of these hours may have to be delegated to the more experienced tester as C.Chapel is still completing onboarding, so, D.Duncan may be brought in to help with the work load.

As 2 working days would usually be 14 hours, testers will have more time to make notes and report any bugs they come across in that time.

Defect Management

When a tester encounters a bug, they will raise a defect on the GitHub Issues page for the project. The tester should assign one of the following priorities:

- **1 - High:** Should be addressed and potentially fixed today

- **2 - Medium:** Should be fixed when possible/time available
- **3 - Low:** Should be fixed eventually

High risk defects will be passed on to Developers as soon as they have been logged, other defects will be looked at on a morning and assessed as to whether there is time to fix them that day.

Completion Criteria

The test plan is completed when all tests listed above have been run and bugs have been logged.

Entry Criteria

Testing will commence when test plan is sent to all testers on this project.

Exit Criteria

Testing will be deemed completed at the end of the week and when all high risk bugs are addressed.

Risks

The following table outlines all of the risks associated with this test plan, and how we intend to mitigate them:

Risk	Mitigation
Any absences in the testing team	D.Duncan can be called in to help, also can see if any other testers on projects that are due to finish later are available if needed.
Product Owner is on holiday for most of the week, it may be hard to get decisions made.	Ensure PO has appointed a person who can make decisions in their absence.
Developers still making last-minute fixes while we test, which could invalidate our testing.	Ensure all last-minute fixes are thoroughly code-reviewed and analysed for scope of impact, with developers doing as much of their own testing as possible.
One variant presents a high priority bug and extra analysis time leads to longer testing session which causes delays	We have assigned extra time for bug logging and other contingencies