**Отчёт по лабораторной работе №4**

**Сценарий 1:**

1. Залогиниться как standard\_user.
2. Сортировать товары по имени (Z to A).
3. Добавить второй товар в корзину.
4. Сортируем по цене (low to high).
5. Добавить третий товар.
6. Сортируем по цене (high to low).
7. Добавить второй с конца.
8. Сортируем по имени (A to Z).
9. Добавить последний товар.

10. Открыть корзину. Открыть описание второго товара в ней.

11. Удалить этот товар. Вернуться в корзину.

12. Открыть описание товара ценой в $9.99.

13. Удалить этот товар. Вернуться в корзину.

14. Нажать Checkout. Ввести в поля “test” “autotest” “123”.

15. Нажать Continue. Проверить, что total цена 25.90.

16. Нажать Finish. Проверить сообщение об успехе. Разлогиниться.

Листинг программы:

// 1. Вход под пользователем standard\_user

function setInputValue(el, val) {

var setter = Object.getOwnPropertyDescriptor(el.\_\_proto\_\_, 'value').set;

setter.call(el, val);

el.dispatchEvent(new Event('input', { bubbles: true }));

}

setInputValue(document.querySelector('#user-name'), 'standard\_user');

setInputValue(document.querySelector('#password'), 'secret\_sauce');

document.querySelector('#login-button').click();

// 2. Добавление товаров

function selectOption(value) {

var sel = document.querySelector('.product\_sort\_container');

sel.value = value;

sel.dispatchEvent(new Event('change', { bubbles: true }));

}

function addToCartByIndex(idx) {

var items = document.querySelectorAll('.inventory\_item');

items[idx].querySelector('.pricebar button').click();

}

selectOption('za');

addToCartByIndex(1);

selectOption('lohi');

addToCartByIndex(2);

selectOption('hilo');

var itemsHilo = document.querySelectorAll('.inventory\_item');

addToCartByIndex(itemsHilo.length - 2);

selectOption('az');

var itemsAz = document.querySelectorAll('.inventory\_item');

addToCartByIndex(itemsAz.length - 1);

// 3. Переход в корзину

document.querySelector('a.shopping\_cart\_link').click();

// 4. Переход к описанию второго товара

var secondCartItem = document.querySelectorAll('.cart\_item')[1];

secondCartItem.querySelector('a').click();

// 5. Удаление товара

document.querySelector('.inventory\_details\_desc\_container button').click();

// 6. Возврат в корзину

document.querySelector('a.shopping\_cart\_link').click();

// 7. Открыть товар с ценой $9.99

const item = Array.from(document.querySelectorAll('.cart\_item'))

.find(el => el.querySelector('.inventory\_item\_price').textContent === '$9.99');

item.querySelector('a').click();

// 8. Удаление товара

document.querySelector('.inventory\_details\_desc\_container button').click();

// 9. Возврат в корзину

document.querySelector('a.shopping\_cart\_link').click();

// 10. Нажатие checkout

document.querySelector('button#checkout').click();

// 11. Ввод данных и нажатие continue

setInputValue(document.querySelector('#first-name'), 'test');

setInputValue(document.querySelector('#last-name'), 'autotest');

setInputValue(document.querySelector('#postal-code'), '123');

document.querySelector('#continue').click();

// 12. Проверка общей суммы

var totalText = document.querySelector('.summary\_total\_label').textContent;

var totalValue = parseFloat(totalText.replace(/[^0-9.]/g, ''));

if(totalValue === 25.90){

console.log('Цена верна: 25.90');

} else {

console.error('Цена неверна:', totalValue);

}

// 13. Нажатие finish

document.querySelector('#finish').click();

// 14. Проверка сообщения об успехе

const headerText = document.querySelector('.header\_secondary\_container').textContent.trim();

if(headerText === 'Checkout: Complete!'){

console.log('Текст верный: Checkout: Complete!');

} else {

console.error('Текст неверный:', headerText);

}

// 15. Логаут

document.querySelector('#react-burger-menu-btn').click();

document.querySelector('.bm-menu-wrap #logout\_sidebar\_link').click();

Все этапы сценария выполняются успешно.

**Сценарий 2:**

1. Залогиниться как standard\_user.

2. Сортировать товары по имени (Z to A).

Листинг программы:

// 1. Вход под пользователем locked\_out\_user

function setInputValue(el, val) {

var setter = Object.getOwnPropertyDescriptor(el.\_\_proto\_\_, 'value').set;

setter.call(el, val);

el.dispatchEvent(new Event('input', { bubbles: true }));

}

setInputValue(document.querySelector('#user-name'), 'locked\_out\_user');

setInputValue(document.querySelector('#password'), 'secret\_sauce');

document.querySelector('#login-button').click();

// 2. Проверка сообщения об ошибке

const errorText = document.querySelector('.error-message-container').textContent.trim();

console.log('Сообщение об ошибке:', errorText);

Все этапы сценария выполняются успешно, сообщение об ошибке появляется.

**Сценарий 3:**

1. Залогиниться как problem\_user.

2. Попробовать выполнить те же действия что и standard\_user в сценарии 1

3. Записать все обнаруженные ошибки.

4. Написать Тест-кейс отчет, ожидаемые результаты (то, что у вас получалось у standard\_user в сценарии 1), полученные результаты (все ошибки, которые словил problem\_user здесь).

Листинг программы: идентичен сценарию 1, за исключением входа под другим логином.

Тест-кейс:

|  |  |  |
| --- | --- | --- |
| **Действие** | **Ожидаемый результат** | **Фактический результат** |
| 1. Пользователь входит в систему под учётной записью problem\_user. | Открывается страница с карточками товаров. Каждая карточка содержит изображение, описание, цену и кнопку добавления в корзину. | Страница загружается, отображаются карточки товаров, но изображения заменены одинаковыми картинками собаки. Описание, цена и кнопки визуально корректны. |
| 2. Пользователь пытается отсортировать товары по имени в обратном алфавитном порядке (Z → A). | Список товаров сортируется по названию от Z до A. | Сортировка не срабатывает, порядок товаров остаётся прежним. |
| 3. Добавление второго товара в корзину. | Выбранный товар появляется в корзине, кнопка «Add to cart» меняется на «Remove». | Поведение соответсвует ожидаемому результату – товар добавляется, кнопка меняет текст. |
| 4. Сортировка товаров по цене от низкой к высокой (Low → High). | Список товаров сортируется от дешёвых к дорогим. | Сортировка не срабатывает, порядок товаров остаётся прежним. |
| 5. Добавление третьего товара в корзину. | Выбранный товар появляется в корзине, кнопка «Add to cart» меняется на «Remove». | Добавить товар не удаётся, кнопка не меняется, товар не появляется в корзине. |
| 6. Сортировка по цене от высокой к низкой (High → Low). | Список товаров сортируется от дорогих к дешёвым. | Сортировка не срабатывает, порядок товаров остаётся прежним. |
| 7. Добавление второго с конца товара в корзину. | Выбранный товар появляется в корзине, кнопка «Add to cart» меняется на «Remove». | Поведение соответсвует ожидаемому результату – товар добавляется, кнопка меняет текст. |
| 8. Сортировка товаров по имени в алфавитном порядке (A → Z). | Список товаров сортируется по названию от A до Z. | Сортировка не срабатывает, порядок товаров остаётся прежним. |
| 9. Добавление последнего товара в корзину. | Выбранный товар появляется в корзине, кнопка «Add to cart» меняется на «Remove». | Добавить товар не удаётся, кнопка не меняется, товар не появляется в корзине. |
| 10. Переход в корзину. | Открывается страница корзины с добавленными товарами. | Страница корзины отображается корректно и содержит выбранные товары. |
| 11. Открытие страницы описания второго товара в корзине. | Открывается страница с информацией о втором товаре из корзины. | Открывается страница товара, которого нет в корзине. |
| 12. Удаление товара из корзины через его карточку. | Товар удаляется, а кнопка «Remove» меняется обратно на «Add to cart». | Удаление невозможно, так как отображаемого товара нет в корзине. |
| 13. Возврат из описания товара обратно в корзину. | Открывается страница корзины с добавленными товарами. | Возвращение в корзину выполняется успешно – корзина отображается корректно. |
| 14. Попытка открыть товар с ценой $9.99. | Открывается страница с информацией о данном товаре. | Открывается страница товара, которого нет в корзине. |
| 15. Переход к оформлению заказа (Checkout) и ввод данных: “test”, “autotest”, “123”. | Отображается форма Checkout: Your Information, все поля корректно заполняются. | Форма открывается, но при вводе текста в поле «Last Name» символы попадают в поле «First Name», где отображается только одна буква. Ввод в поле «Last Name», таким образом, невозможен. |
| 16. Нажатие на кнопку Continue и проверка общей суммы (total = 25.90). | Открывается страница Checkout: Overview, и на ней отображается итоговая сумма товаров 25.90. | Переход невозможен из-за ошибки ввода в поле «Last Name», которое остаётся пустым. |
| 17. Попытка завершить заказ нажатием кнопки Finish и проверка сообщения об успешной покупке. | Появляется страница Checkout: Complete! с надписью «Thank you for your order!». | Переход не выполняется в следствие невыполнимости предыдущего шага. |
| 18. Выход из аккаунта. | Пользователь выходит из системы. | Выход выполняется успешно. |

**Сценарий 4:**

1. Залогиниться как performance\_glitch\_user.

2. Он залагал, ждем.

3. Открыть описание первого товара, добавить в корзину.

4. Вернуться в список товаров.

5. Опять залагал….

6. Сортировать по цене (low to high).

7. Ждем пока отлагает… добавляем товар за 49.99 в корзину.

8. Надоели лаги, поэтому разлогиниваемся :)

Листинг программы:

function setInputValue(el, val) {

var setter = Object.getOwnPropertyDescriptor(el.\_\_proto\_\_, 'value').set;

setter.call(el, val);

el.dispatchEvent(new Event('input', { bubbles: true }));

}

var wait = ms => new Promise(r => setTimeout(r, ms));

// 1. Логин

setInputValue(document.querySelector('#user-name'), 'performance\_glitch\_user');

setInputValue(document.querySelector('#password'), 'secret\_sauce');

document.querySelector('#login-button').click();

// 2. Подождать, пока страница "отлагает"

await wait(2000);

// 3. Открыть описание первого товара и добавить в корзину

document.querySelectorAll('.inventory\_item')[0].querySelector('a').click();

await wait(2000);

document.querySelector('.inventory\_details\_desc\_container button').click();

// 4. Вернуться к списку товаров

window.history.back();

await wait(2000);

// 5. Сортировать по цене (low-high)

var sel = document.querySelector('.product\_sort\_container');

sel.value = 'lohi';

sel.dispatchEvent(new Event('change', { bubbles: true }));

await wait(2000);

// 6. Добавить товар стоимостью $49.99

var items = Array.from(document.querySelectorAll('.inventory\_item'));

var target = items.find(i => i.querySelector('.inventory\_item\_price').textContent === '$49.99');

if (target) target.querySelector('.pricebar button').click();

// 7. Разлогиниться

await wait(2000);

document.querySelector('#react-burger-menu-btn').click();

await wait(2000);

document.querySelector('.bm-menu-wrap #logout\_sidebar\_link').click();

**Сценарий 5:**

1. Зайти как error\_user.

2. Добавить все товары в корзину.

3. Проверить все ли добавились. Записать ошибки.

4. Убрать все добавленные товары.

5. Проверить удалось ли убрать. Записать ошибки.

6. Перейти в корзину нажать Checkout.

7. Попробовать ввести в поля “test” “autotest” “123”.

8. Нажать Continue. Нажать Finish.

9. Если не все сработало, записать ошибки.

10. Написать Тест-кейс отчет, ожидаемые результаты (то, что у вас получалось у standard\_user в сценарии 1), полученные результаты (все ошибки, которые словил error\_user здесь).

Листинг программы:

function setInputValue(el, val) {

var setter = Object.getOwnPropertyDescriptor(el.\_\_proto\_\_, 'value').set;

setter.call(el, val);

el.dispatchEvent(new Event('input', { bubbles: true }));

}

// 1. Вход в пользователя error\_user

setInputValue(document.querySelector('#user-name'), 'error\_user');

setInputValue(document.querySelector('#password'), 'secret\_sauce');

document.querySelector('#login-button').click();

// 2. Добавить товары в корзину

const addButtons = document.querySelectorAll('.inventory\_item .pricebar button');

addButtons.forEach(btn => btn.click());

console.log(`Добавлено товаров: ${addButtons.length}`);

// 3. Проверить, все ли добавились

const removeButtons = document.querySelectorAll('.inventory\_item .pricebar button');

const addedCount = Array.from(removeButtons).filter(b => b.textContent === 'Remove').length;

console.log(addedCount === addButtons.length

? 'Все товары успешно добавлены'

: `Добавлены не все товары: ${addedCount}/${addButtons.length}`);

// 4. Убрать все добавленные товары

removeButtons.forEach(btn => {

if (btn.textContent === 'Remove') btn.click();

});

// 5. Проверить, всё ли убрано

const remaining = Array.from(document.querySelectorAll('.inventory\_item .pricebar button'))

.filter(b => b.textContent === 'Remove').length;

console.log(remaining === 0

? 'Все товары удалены'

: `Осталось неубранных: ${remaining}`);

// 6. Перейти в корзину и нажать Checkout

document.querySelector('a.shopping\_cart\_link').click();

document.querySelector('button#checkout').click();

// 7. Ввести test, autotestm 123

setInputValue(document.querySelector('#first-name'), 'test');

setInputValue(document.querySelector('#last-name'), 'autotest');

setInputValue(document.querySelector('#postal-code'), '123');

// 8. Попробовать нажать Continue и Finish

document.querySelector('#continue').click();

const finishBtn = document.querySelector('#finish');

if (finishBtn) {

finishBtn.click();

console.log('Finish нажата');

} else {

console.warn('Кнопка Finish недоступна');

}

Тест-кейс:

|  |  |  |
| --- | --- | --- |
| **Действие** | **Ожидаемый результат** | **Фактический результат** |
| 1. Вход в пользователя error\_user | Открывается страница с карточками товаров. Каждая карточка содержит изображение, описание, цену и кнопку добавления в корзину. | Страница загружается полностью корректно. |
| 2. Добавление всех товаров в корзину | Все товары добавляются в корзину, а кнопки «Add to cart» меняются на «Remove». | Добавлено только 3 из 6 товаров, у части товаров кнопки изменились, но некоторые товары остались без изменений и не попали в корзину. |
| 3. Удаление всех товаров из корзины | Все добавленные товары удаляются, а кнопки заменяются с «Remove» обратно на «Add to cart». | Все товары остались в корзине, подписи на кнопках не изменились. |
| 4. Переход к оформлению заказа | Открывается страница Checkout с полями для ввода. | Страница Checkout открылась корректно. |
| 5. Ввод данных “test”, “autotest”, “123” в поля формы. | Все данные корректно вводятся в поля для ввода. | При вводе в поле «Last Name» выводится ошибка: «Cannot read properties of undefined (reading ‘value’)». Таким образом, поле недоступно для ввода и данные в него не записались. |
| 6. Переход к Checkout. | Открывается страница Checkout: Overview. | Страница открылась, но одно из полей («Last Name») осталось пустым. |
| 7. Завершение оформления заказа. | После нажатия на кнопку Finish открывается страница «Checkout: Complete!» с сообщением об успешном заказе. | Кнопка не сработала, вывелась ошибка: «C.cesetRart is not a function». Завершить заказ не удалось. |