## Executive summary:

ShowPo is a fast-moving midsize fashion eCommerce company which has been replatforming every 2 to 3 years as their business grows. To monitor and manage the stability of their website, they create a regression test suite at each replatforming. In the attached document, I outline a good cross-section of what I think would be how they test reliability and functionality, as well as dip my toes into the other elements of testing.

This piece is not a complete set of the tests that would be included in their regression test suite. I don’t cover all negative test cases, and I have only included customer-facing functions. However, I hope to provide a good cross-section which demonstrates to you how I think as a tester. While automated tests don’t test everything, the aim of a good regression test suite is to give confidence that the program works as it should.

## Introduction

I’ve been following ShowPo since 2011. Showpo is a local Sydney company which has grown to become one of Australia’s leading fashion companies and eCommerce properties. To me, they look like the perfect case study to use as an example of how I can apply my software testing skills to any eCommerce product. I used recent articles and BuiltWith to gain an understanding of their technology stack, and help me decide which tools might be the most appropriate for them to use. I am making the assumption that ShowPo uses the agile approach to software development, based on interviews given by the founder. In writing up this case study, I opted not to talk to anyone at ShowPo. It is based purely on my experience of their site and of eCommerce itself. In a previous life, I worked as an eCommerce coordinator where I first developed skills in user acceptance testing, exploratory testing, and ad-hoc testing. As part of my self-study for my testing career I am learning the other forms of testing and how to apply them. I do a general overview of some of them here.

With this case study I aim to answer, ‘what would you test?’ and ‘how would you test it?’ during the relaunch of an eCommerce site to create a master test plan - a regression test suite used throughout the life of an eCommerce company’s use of a platform before they move onto more something suitable for their next stage of growth. It is created in the scoping out and diagramming requirements part of the initial sprint, in the very earliest stages of the Agile Software Development Life Cycle. This would place ShowPo at at least TMMi Level 3.

DISCLAIMER: This piece is not a complete set of the tests that would be included in their regression test suite. I don’t cover all negative tests, and I have only included customer-facing functions. However, I hope to provide a good cross-section which demonstrates to you how I think as a tester.

Test Approach:

Using my personal knowledge of the eCommerce domain, I aimed to write comprehensive functional test cases.

My goal was to follow the [80/20 rule](https://www.scottlawrence.org/2016/12/13/best-practices-software-testing/), with 80% of the test scenarios covering failure cases, and 20% covering success cases. I started with the **happy path**, then wrote **negative tests**. To find these cases first I explored **boundary/range testing**, and **null testing**. Many of my E2E tests would end up having multiple asserts which violates the Single Responsibility Principle. In these cases, I run elements of these bigger tests separately, to better isolate what might be causing a bug. Of course, there are also **edge cases**, where a non-typical customer will enter data that should be accepted, but often isn’t. These assumptions about users, through testing mostly the happy path of user experience, can negatively impact UX, usability, and **accessibility**. As well as these edge cases, I also explored whether ShowPo had catered for **empty states**, where content had not yet been added or could not yet be found but a blank state would look rough and unprofessional. Additionally, I tested a few **user expectations** and **claims** in the Help & Contact section.

Generally, only custom code added on-top of framework generated code needs testing, except in the cases where the core behaviour is changed.

I have used a variety of Arrange Act Assert TDD syntax and Gherkin BDD syntax to describe how to test these cases.

## Test Strategy

### Test level

* Integration tests

Environment requirements

* Linux Ubuntu 16.04 or Mac OS Catalina

### Scope

* Functional testing
  + This test plan targets the main user journeys. The key focus of the test plan is functionality and reliability testing - two of the six measures in ISO 9126.
    - Finding outfits to choose from for a party
    - Working out if the outfit can get to her in time
    - Purchase items using a coupon code
    - Finding out the cost of shipping before putting anything in the cart
    - Looking up ShowPo policy
    - Purchasing an outfit
    - Saving credit card details for future purchases
    - Saving home & work shipping addresses
    - Creating an account for an easy way to track all ShowPo orders
    - Student discounts
    - Returning an item
    - Purchase items using a gift card
* Reliability testing
  + This test plan covers some situations that ShowPo has over the course of the year
    - Planning for expansion: Soak testing
    - Major sales: Stress & spike testing

### Out of scope

* Efficiency testing
* Maintainability testing
* Portability testing
* Usability testing
* Some user journeys
  + Creating gift cards
  + Getting notified about a product that is out of stock
  + Admin actions

### Assumptions

* Aside from the load and data, the test environment matches the production environment.
* Test data to check the assumptions of the tests has been added into the test environment.
* The tests are tagged by feature areas so they can be run separately during development.
* Unit tests are added to the CI/CD pipeline.
* Smoke tests are run for each release candidate.
* Regression tests are run daily, overnight if necessary.
* Tests are run in parallel in parallel

Defect Management

1. Issues are discovered by testers and developers.
2. Defects are categorized for each product risk category [by policy.](https://www.mountaingoatsoftware.com/blog/defect-management-by-policy-a-fast-easy-approach-to-prioritizing-bug-fixes) They are judged by the percentage of transactions affected. Sentry data is analysed to get this information. Likelihood & Severity.
3. Defects are reported to the development team through GitHub issues or JIRA.
4. Defects are fixed by the developers
5. Defects are verified fixed by testers
6. Issue is closed
7. Defect report at end of sprint

### Deliverables

* Automated tests report
* Defects reports

### Testing Tools

* Tools are chosen for each task to keep tests easier to maintain, and easier to write. Ad-hoc and exploratory testing would happen during the development phase. To view the test automation framework please see the [appendix](https://docs.google.com/document/d/16PQdYz-D92D-0YjdOkU5i-CSfFO4AXUTjTn8LoTQ6wY/edit).
* As a fast moving company that does replatform every 2 to 3 years as they grow, automating testing where sensible enables them to move faster, while keeping a firm check on stability. Tests that are simpler and take a shorter amount of time to write are preferred, as with all elements of the testing process.
* Manual for some functional tests and mobile responsiveness tests
* Selenium for most functional tests
* Applitools & BrowserStack for visual testing and cross browser testing
* Axe, and Pa11y-dashboard for first-pass accessibility testing
* Postman for API testing, and search/list type functionality
* Salesforce [recommends](https://documentation.b2c.commercecloud.salesforce.com/DOC3/index.jsp?topic=%2Fcom.demandware.dochelp%2FLegacyDevDoc%2FSiteGenesisSetup.html) using Selenium server for what they define as unit tests, and Webdriver for multiple browser and headless testing, where the user interface is not required. They also recommend Mocha for the test framework.
* Many of the core features will have had system tests already written by Salesforce in the development of the platform.
* For reliability testing, use a combination of testing and monitoring.
* Google Analytics for monitoring what browsers + device combinations to test
* Datadog for instrumentation
* Sentry for application monitoring and real-time error tracking
* Gatling/Flood/k6 for load testing

*to briefly touch on one of the other elements of software testing*

* Linting, for example prettier, can assist in testing small aspects of code maintainability

### Risks and Risk Management

* Taking the [PRISMA](http://www.erikvanveenendaal.nl/NL/files/e-book%20PRISMA.pdf) method, product risks are below. Risky features can be defined by the potential loss of income and reputation if they should fail. Features which are less crucial, but are nonetheless highly complex, are also risky, but carry less priority in a risk matric Areas that are changed frequently with time pressures like promotions, are also risky. Another way that I used to see what the company considered most risky was to look at what features were prioritised in the header navbar, particularly on mobile view, as these are the features found to be most important to customers outside the obvious after 8 years of conversation and monitoring.

|  |  |
| --- | --- |
| Priority 1 product risks: catastrophic  * Payments * Cart * Checkout * Transactional email * Internationalisation * Catalog search * Log in / Log out | Priority 2 product risks: damaging  * Returns * Order tracking * Contact form * Help Centre * Promotions |
| Priority 3 product risks: hindering  * Shipping times check * Wishlist * Mailing list form * Search * Create account | Priority 4 product risks: annoying  * Blog * Edits * Size Guide |

### Suspension criteria

* Issues with the test environment. Could indicate other issues.
* The number of critical defects.
* The number of non-reproducible functional defects for high priority product features is above 2.

### Exit criteria

* No outstanding high priority functional defects (of P1 and P2)

Legend

|  |  |
| --- | --- |
| Negative test |  |
| Positive test |  |

# Reliability

It would be remiss to write a theoretical master test plan for a powerhouse like ShowPo and not talk about performance & reliability.

ShowPo regularly runs big email marketing campaigns, sales, and they get a lot of press, creating stress loads on their infrastructure. They also have a not insignificant number of regular daily users!

To get the numbers for the reliability tests, I read news articles about Showpo that included financial figures.

I used $100 million in projected revenue for 2020 and a 8% conversion rate as my guide to this number to find a baseline, as well as an average order value of $70. A 4% is more typical. To measure performance requirements for that, simply double the below numbers.

The typical ecommerce fashion and apparel website, according to Fireclick Index, has a 7.2% conversion rate.

SimilarWeb say they get 1.01 million visits a year, with 5.8 pages a visit, an 8 minute long visit, and a 49.8% bounce rate. I think 1.01 million visits is a little of an underestimation, but the rest of the numbers look about right.

The Performance SLAs I have used are based on e-commerce industry benchmarks.

As their main European market is the UK, I assume that the bulk of their traffic is from 18 hours of the day, starting from NZ waking hours rolling over to the UK early evening hours.

Reliability testing is new to me, so in this section I’m just going to throw out a few ideas based on the calculations above.

Expected outcome is based on the ecommerce expectation of five nines uptime, especially for a company like ShowPo.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **What to test** | **Expected outcome** | **How to test** |
| Smoke | Home page load  2 concurrent users  Mimic real user combinations. Configure time zones, IP addresses and locations, network speeds, etc, to mimic the environment the target audience is in.  Test top-of-the-funnel conversions on mobile   * Mailing list signup * Unidays sign up | *Measures performance with a minimal number of users*  0 errors  100% of requests finish within 1 second. | Gatling or k6 |
| Load - authenticated  Typical load | 38,051 users over 24 hours, the majority over 18 hours.  10,651 adds to cart.  3,044 transactions.  Averaging 3200 concurrent users.  400 for an hour  700 for an hour  800 for an hour  1000 for an hour  1580 for an hour  2110 for an hour  2330 for an hour  2440 for an hour  2530 for an hour  2800 for an hour  3550 for an hour  3400 for an hour  2500 for an hour  2550 for an hour  2530 for an hour  2000 for an hour  1580 for an hour  1580 for an hour  1300 for an hour  800 for an hour  600 for an hour  500 for an hour  300 for an hour  200 for an hour | *Measures performance of a system under an expected load*   * 99% of requests should finish within 5000ms. * 95% of requests should finish within 1000ms. * 99% users should be able to login successfully on the first try | Gatling or k6  Create fake data with fakerjs to put into the Mocha test suite  Home page & catalog page  60% mobile  40% desktop |
| Load - unauthenticated  Typical load | 38,051 users over 24 hours, the majority over 18 hours.  10,651 adds to cart.  3,044 transactions.  Averaging 3200 concurrent users.  400 for an hour  700 for an hour  800 for an hour  1000 for an hour  1580 for an hour  2110 for an hour  2330 for an hour  2440 for an hour  2530 for an hour  2800 for an hour  3550 for an hour  3400 for an hour  2500 for an hour  2550 for an hour  2530 for an hour  2000 for an hour  1580 for an hour  1580 for an hour  1300 for an hour  800 for an hour  600 for an hour  500 for an hour  300 for an hour  200 for an hour | * 99% of requests should finish within 5000ms. * 95% of requests should finish within 1000ms. | Gatling or k6  Create fake data with fakerjs to put into the Mocha test suite  60% mobile  40% desktop |
| Load - authenticated  Peak load | Gradually ramps up to 7000 concurrent users | * 99% of requests should finish within 5000ms * 95% of requests should finish within 2000ms. | Gatling or k6  Create fake data with fakerjs to put into the Mocha test suite  60% mobile  40% desktop |
| Load - unauthenticated  Peak load | Gradually ramps up to 7000 concurrent users | * 99% of requests should finish within 5000ms * 95% of requests should finish within 2000ms. | Gatling or k6  Create fake data with fakerjs to put into the Mocha test suite  60% mobile  40% desktop |
| Stress | Ramping up to 90,000 users in 9 hours, or double what I project ShowPo’s average daily user number is.  Ramping up to 70,000 users searching through at least 3 pages in these 9 hours  Ramping up to 160,000 add to carts in 9 hours.  Ramping up to 4,000 transactions in 9 hours  7,500 transactions in 9 hours, gradually ramping up to 1,500 transactions an hour. This would be around two times more than ShowPo’s typical daily number of transactions.  Three locations (from SimilarWeb data & Alexa to simulate Google Analytics data.)   * AU * US * NZ or Canada   See how many users visited the site in a second after an email campaign or news article.  Run GET requests on the top 5 most visited pages | *Overloads a system in order to find the breaking point. Helps determine how quickly the auto-scaling mechanisms react to increased load, and if there are any failures during the scaling events.*  The auto-scaling mechanism reacts quickly to the increased load, and then down scales just as well.   * 98% of requests should finish within 5000ms * 95% of requests should finish within 1500ms | Gatling or k6  Create fake data with fakerjs to put into the Mocha test suite  60% mobile  40% desktop  Most visited pages - homepage, sale, dresses, new arrivals. |
| Spike | 12,000 concurrent users.  60,000 home page loads in  10 minutes.  20,000 adds to cart in 20 minutes  2,000 transactions in 10 minutes | *In UAT or staging environment. A sudden instant surge of traffic.*  For ShowPo customers would expect either excellent or good system performance. It is either not degraded during the surge of traffic, or the response time is slower, but the system does not produce any errors.   * 99% of requests should finish within 5000ms. * 95% of requests should finish within 3000ms. | Flood or Gatling or k6  Create fake data with fakerjs to put into the Mocha test suite  65% mobile  35% desktop |
| Soak | 80% capacity of system  1 hour test  8,000 concurrent users | *uncovers performance and reliability issues stemming from a system being under pressure for an extended period.*   * Verify that the system doesn't suffer from memory leaks * Verify that expected application restarts don't lose requests. * Find bugs related to race-conditions that appear sporadically. * Make sure your database doesn't exhaust the allotted storage space and stops. * Make sure logs don't exhaust the allotted disk storage. * Make sure the external services ShowPo depends on don't stop working after a certain amount of requests are executed. * 99% of requests should finish within 5000ms. * 95% of requests should finish within 3000ms. | Flood or Gatling or k6  Create fake data with fakerjs to put into the Mocha test suite  60% mobile  40% desktop |

# Mobile responsiveness

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Page Load | Buttons are a comfortable size  No elements are squished  No elements overflow into the sides of the page | Browserstack |

# Cross Browser Compatibility

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Page load | Looks similar on any desktop browser as it does on Chrome. If it looks different, it fails gracefully.  Dropdowns / buttons / etc retain the same functionality, if not the same look, on all targeted browsers to the right. | BrowserStack  Focus on browser/device combinations   * favoured by end-users * Have the highest usage trends in your target market * Correlated with large drops on the conversion funnel |

# Accessibility

Accessibility is not a government requirement of businesses in Australia. For the purpose of this exercise I have chosen WCAG 2.0 to represent ShowPo’s diverse customer base.

|  |  |
| --- | --- |
| **Description of activity** | **How to test** |
| Provide text alternatives for non-text content | aXe linter |
| Provide captions and other alternatives for multimedia | aXe linter |
| Make it easier for users to see and hear content | aXe linter  ChromeVox |
| Make all functionality available from a keyboard | Manual |
| Give users enough time to read and use content | Manual  Check modal alerts don’t flicker on and off. |
| Do not use content that causes seizures | Manual  GIFS are part of ShowPo’s brand identity at this point. So it is important to check they don’t “flash”, are slower, with less contrast. |
| Help users navigate and find content | Manual |
| Make text readable and understandable | Manual  Check text isn’t too small. Check that the reading age level is as low as possible. |
| Make content appear and operate in predictable ways | Manual. Check for use of well-known UI design patterns. |
| Help users avoid and correct mistakes. | Manual. Inspect alert messages for specificity and readable language. |
| Maximize compatibility with current and future user tools. | Manual |
| Links must have discernable text | aXe linter |
| Viewport doesn’t have zooming and scaling disabled | aXe linter |
| Elements have sufficient colour contrast | aXe linter |
| Buttons have discernable text | axe |

# Internationalisation

Non-AU customers made up 1/3 of ShowPo’s revenue as of late 2019.

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Click dropdown in top Navbar and click $NZ | Website URL changes to <https://www.showpo.com/nz/> | GIVEN I am on the home page  WHEN I click the currency dropdown in top Navbar  AND click $NZ  THEN the new URL contains showpo.com/nz/ |
| Click dropdown in top Navbar and click €EU | Price display changes to € | Selenium  GIVEN I am on the dresses catalog page  WHEN I am an item to cart  AND click on the cart icon  THEN the totals table has the € currency symbol. |
| Click dropdown in top Navbar and click $NZ | Default shipping options country changes to New Zealand | Selenium  Add item to cart  Click on cart page  See if active country in shipping options dropdown is New Zealand |
| Click dropdown in top Navbar and click $NZ | Promotion changes to expected NZ promotion | Click dropdown in top Navbar and click $NZ  Find the promotion element  Assert it matches expected statement |
| Set location of browser as somewhere in Europe with VPN | When loading showpo.com, it redirects to showpo.com/eu/ | Selenium Headless Browser  Load showpo.com  Wait  Assert that showpo.com/eu/ is part of URL |
| Use Arabic characters in name fields | Success | Unit test |
| Use Cyrillic characters in name fields | Success | Unit test |
| Use umlauts in name fields. | Success. | Unit test |
| Use umlauts in address fields. | Success. | Unit test |
| Use unicode in name fields. | Success | Unit test |
| Use apostrophes in name fields. | Success | Unit test |
| Use apostrophes in address fields. | Success | Unit test |
| Use accents in address fields. | Success | Unit test |
| Use accents in name fields. | Success | Unit test |

# Functional tests

## Cart

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Add item to cart | Item is added to cart | Selenium  Go to category page  Click //\*[@title="Add to bag"][1]  Assert order total is > $0. |
| Add item to cart | “It’s in the bag!” pops up | Selenium  Go to category page  Click //\*[@title="Add to bag"][1]  Wait for popup item element to appear  Assert text in popup |
| Remove some items from cart | Product removed from cart and total sum decreased correspondingly. An alert appears ‘Are you sure you want to remove this item from your bag?’ | Selenium.  Click the remove item button  Assert the existence of modal with “Are you sure you want to remove this item from your bag?” |
| Clicking on cart when empty | Alert that it is empty and button to continue shopping | Selenium  Assert the existence of an alert message that the cart is empty. |
| Increase the quantity of the item in the cart | Quantity increases, as does price and tax correspondingly | Selenium  Go to quantity dropdown of first item in cart  Enter ‘3  Assert that the quantity of that item is 3 |
| Add the same item to cart multiple times | Product still only appears once in cart, but quantity increases | Selenium  Click add to cart for first size available in catalog list  Add the site element again  Go to cart  Assert that the quantity of the item is 2 |
| Add multiple items of different types | For each item added, we should see a corresponding name, image, and price and total price of all items. | Selenium  Go to category page  Click //\*[@title="Add to bag"][1]  Click //\*[@title="Add to bag"][1]  Click //\*[@title="Add to bag"][1]  These will be three separate items so there should be three separate products in the cart  Assert there are three separate products in the cart |
| Click on an item in the cart | See more information about the product with a redirection to the product page | Unit test |
| Add item(s) to the cart, close the browser and reopen the same site | Cart still has items. | Manual Add item(s) to the cart  Close the browser  Open the browser  Reopen the same site  Assert the totals field exists |
| Remove all items from the cart | No items in cart, grand total is zero. | Unit test |
| Click ‘Go to checkout’ | Directed to checkout page | Selenium  Click go to checkout  Assert URL is for shipping page |
| Edit Shipping address on payment page | Return to shipping page | Selenium  Click Edit Address button  Wait  Assert URL contains shipping#shipping |
| ‘Apply a coupon code’ in cart | Apply coupon code | Selenium  Save current subtotal as variable  Click ‘apply a coupon code’ button  Find textbox  Add coupon code  Click enter  Assert new subtotal is lower than previous subtotal |

## Checkout

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Choose pickup with same-day time frame when cart value is above 50 | Shipping price is added and total increased | Unit test |
| Choose pickup with 5 day time frame when cart value is above 50 | Shipping price is removed and total reduced | Unit test |
| Choose delivery with same-day time frame when cart value is above 50 | Shipping price is added and total increased | Unit test |
| Choose delivery with 5 day time frame when cart value is above 50 | Shipping price is added and total increased | Unit test |
| Add letters in the phone number field, all else valid. | *Negative test*  Eorror message | Unit test |
| Add valid phone number, but with spaces in it for human readability | *Edge case - Success [*Phone number is ambiguous, it could be both a landline or mobile phone, so there are no character limits.] | Unit test |
| Apply valid coupon code | Remove correct amount from the balance Add an extra line to the order details with a minus. | Unit test |
| Apply invalid coupon code | *Negative test*  Upon clicking apply, alert that the card is invalid. | Unit test |
| Checkout as guest | Option is available on <https://www.showpo.com/checkout-login/> screen | Selenium  Add item to cart  Go to cart  Click go to checkout  Enter email address  Click next step button  Assert that link with ‘checkout as guest’ option appears |
| Product with auto-discount is added to cart | It is automatically discounted on checkout | Unit test |
| Add a invalid gift card | *Negative test*  Upon clicking balance or apply, alert that the card is invalid. | Unit test |
| Add an valid gift card | Remove the amount left on the gift card from the balance Add an extra line to the order details with a minus. | Unit test |
| Click ‘Remember this card’ | Card details are saved to account | Unit test |
| Click ‘Pay now with afterpay’ | Redirected to afterpay website | Unit test |
| Click ‘Pay now with PayPal’ | Redirected to PayPal website | Unit test |
| Click ‘complete order’ after all details are successfully added. | Directed to order complete page Order processing email received. | Unit test with test data or mocking |
| Posting to a regular address for Qatar, UAE | *Negative test - claims / user expectations* | Unit test |
| Posting to PO boxes for Qatar, UAE |  | Unit test |
| Posting to PO Box for today, tomorrow, or Saturday | *Negative test - claims / user expectations* | Unit test |
| Posting to PO Box in 5 days |  | Unit test |
| PO Box order of bridal gowns | *Negative test - claims / user expectations*  Entering this text in the shipping address when a bridal gown is in the cart creates an error alert. | Unit test |
| private bag order of bridal gowns | *Negative test - claims / user expectations*  Entering this text in the shipping address when a bridal gown is in the cart creates an error alert. | Unit test |
| parcel locker order of bridal gowns | *Negative test - claims / user expectations*  Entering this text in the shipping address when a bridal gown is in the cart creates an error alert. | Unit test |

## Payments

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Check different payment types can pay when details entered correctly | Successful payment through Card (probably Stripe, PayPal, and AfterPay (after approval) | Unit test |
| Check credit card details are stored correctly if stored. | PCI compliance. | Database test  There should only be the last 4 characters stored. |
| Card with invalid details | *Negative test*  Alert message of failure | Unit test |
| Card with insufficient balance | *Negative test*  Alert message of failure | Unit test |
| Card with missing details | *Negative test*  Alert message of failure | Unit test |
| Card with invalid expiry date | *Negative test*  Alert message of failure | Unit test |
| Card with invalid CVV | *Negative test*  Alert message of failure | Unit test |

## Sessions

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Session time | Session time is as set in administration | Postman |
| While logged out, go to <https://www.showpo.com/accounts> | Redirect to <https://www.showpo.com/login/?rurl=1> | Selenium  Click <https://www.showpo.com/accounts>  Wait  Assert the current url is <https://www.showpo.com/login/?rurl=1> |

## Transactional emails

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Order processing and order received emails | Email is sent to customer and admin when order is made | Unit test with mocking |
| Wishlisted product email | Sent when product status changes [I don’t know if that’s what wishlist does. That’s what I would expect it to do] | Unit test with mocking |
| Cart abandonment emails | Appears in customer inbox after a certain number of hours/days | Unit test with mocking |
| Order cancelled email | Email is sent to customer and to admin | Unit test with mocking |

### Order tracking

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Status: order not picked & packed yet | Email: processing order. | Unit test with mocking |
| Tracking available email | Email is sent to customer when shipping code is made | Unit test with mocking |
| Status: Received by Courier | Email: Received by courier | Unit test with mocking |
| Status: Out for Delivery | Email: Out for delivery | Unit test with mocking |
| Status: Completed | Email: Delivery complete | Unit test with mocking |

Contact company through email

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Use Contact us form | All fields with data entered are sent in email to Zendesk | Unit test with mocking |
| Use “mailto” link | Device default email client is opened up | Selenium |
| Use UI form to join mailing list | Email is added to the relevant list | Check email for ‘thankyou for subscribing’ email |
| Forget @ in email address in email subscription form | *Negative test*  Alert message: “Please provide a valid email Id” | Enter email wrong  Click ‘GO!’ |
| Forget . in email address in email subscription form | *Negative test*  Alert message: “Please provide a valid email Id” | Enter email wrong  Click ‘GO!’ |
| Click ‘Go!’ in email subscription form without entering email address | *Negative test*  Alert message: “Please provide a valid email Id” | Unit test |
| Forget . in email address in contact form | *Negative test*  Alert message: “Please enter a valid email address” | Unit test |
| Forget @ in email address in contact form | *Negative test*  Alert message: “Please enter a valid email address” | Unit test |
| Click ‘start chat’ without entering information in any fields in contact form | *Negative test*  Alert message: “Please enter a valid name/email address/message” | Unit test |

## Log in / Log out / Create account

### Sign up

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Sign up with email address that already has an account | *Negative test*  Alert message: ‘Email already exists’ | Unit test |
| Sign up with any compulsory field missing | *Negative test*  Alert message: ‘Please fill in this field’ | Unit test |
| Sign up through OAuth when logged into Facebook | *Happy Path*  A new account registered through OAuth FB | Selenium  Go to the Signup tab on the Login page  Click ‘Sign up with FB’  A prompt asking for access to FB  Assert that you have been redirected to the account page and therefore a ShowPo account has been created. |
| Sign up through OAuth when not logged into Facebook | *Edge Case*  A new account registered through OAuth FB | Selenium  Go to the Signup tab on the Login page  Click ‘Sign up with FB’  A prompt asking you to log in to FB  A prompt asking for access to FB  Assert that you have been redirected to the account page and therefore a ShowPo account has been created. |
| Sign up with a two character first name | *Edge Case*  Success. | Unit test |
| Sign up with a 100 character first name | *Edge Case*  Success. | Unit test |
| Sign up with a two character last name | *Edge Case*  Success. | Unit test |
| Sign up with a 100 character last name | *Edge Case*  Success | Unit test |
| Sign up with first/last name that has leading/trailing spaces | *Success*  Leading / Trailing spaces are deleted | Sign up  Check DB record for that first/last name has no leading or trialing spaces. |
| Cannot perform SQL injections | *Negative test*  Failure | DROP TABLE users  DROP TABLE customers  TRUNCATE TABLE users  TRUNCATE TABLE customers |
|  |  |  |

### Log in

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Log in with non-matching email and password fields | *Negative test*  Error message. | Selenium  GIVEN I am on the Login screen  WHEN I enter email address with non-matching password  AND I submit the form  THEN I get a refresh of the page with error messages |
| Log in with the correct details | *Happy path*  Succeed and the account page | Selenium  GIVEN I am on the Login screen  WHEN I enter email address with a matching password  AND I submit the form  THEN I get a redirect to a page with URL ‘/account/’ |

### Forgot password

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Click ‘forgot password’ | Go to page where I can enter email address to get a reset password email | Selenium  GIVEN I am not the Login screen  WHEN I click the ‘forgot password’ link  THEN I am ON the forgot password modal |
| Enter email address connected to account in in ‘forgot password’ page | Receive ambiguous alert message  “If there is an account associated with \_\_\_@\_\_\_\_.\_\_\_, you will receive an email with a link to reset your password”  Receive reset password email | Unit test |
| Enter email address not connected to account | *Negative test*  Receive ambiguous message, but no email because there no account  “If there is an account associated with \_\_\_@\_\_\_\_.\_\_\_, you will receive an email with a link to reset your password” | Unit test |

### Log out

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Click log out | Log out of account | Selenium  GIVEN I am logged into my account  WHEN I click ‘Log out’  THEN the account link is now /login |

## Membership features

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Review order history | All orders by customer using email address of account appear in order history section | Automated Database Testing  Selenium |
| Add shipping address in address book | Shipping address is added | Selenium  GIVEN I am on the shipping address page  WHEN I add a valid shipping address  AND click ‘enter’  THEN I can see it on the page |
| Customer cancels order from account page | Order is updated in account page An email is sent to admin | Selenium  GIVEN I am on account admin page  WHEN I click the button to cancel my newest order  THEN the order number is blurred out |

### Student discount

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Join UNiDAYS if not already a member | Page changes to the verify account page | Selenium  GIVEN I am on Join UNiDays page  WHEN I enter email in Email field  AND Enter email in Confirm email field  AND Enter password in Password field  AND Enter password in confirm Password field  AND Enter gender in Gender field  AND Click Confirm over 18 & comply etc box  AND Click Join now  THEN I am on the verify account page |
| Verify account | Verified account | Selenium  Got to page  Enter first name  Enter last name  Search for and enter institution name  Enter year of study  Enter course length  Click ‘Continue’ |
| Send ticket with information on institution | Ticket sent | Selenium  Go to page  Enter first name  Enter last name  Enter institution name  Enter personal institution email address  Enter it again  Click ‘Submit ticket’ |
| Log in with verified account | Log in | Selenium  Click ‘10% student discounts’  Enter email in Email field  Enter password in Password field  Click sign in button  Assert the url is the account page |
| 10% student discount applied | 10% student discount applied | Sign in as UNiDAYS Verified account  Add items to cart  Assert that the discount total is 1/9 of total price. |

## Product detail page

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Information appears | Necessary fields appear on product detail page   * Image or images of the product * Price of the product * Product specifications * Reviews * Check out options * Delivery options * Shipping information * In stock/Out of stock * Multiple color or variations options * Breadcrumb navigation for the categories | Check if titles for each category appear in the code. Selenium You could also use the API to see what information comes out, but the front-end team also decides what fields from those to use. |
| Multiple sizes - Select size and add to bag | Adds product to bag | Selenium  Select first available size and click  Save number of button as variable  Click ‘Add to bag’  Assert number of button matches size field in cart |
| Single option - Add item to bag | Adds product to bag | Selenium  Go to Beauty catalog page  Click ‘Add to bag’  Assert there is one item in the cart |
| Click left and right buttons of product image section | New photos are shown in main view | Selenium. |
| Click on product images | See them in the main view | Selenium. |

## Promotions

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| View promotions | Promotions don’t include out-of-stock items | Postman. Check the API request / SQL query |

## Catalog page

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| View the next page via pagination. | View the next page via pagination. | Selenium.  Click link within div class=”pagination” with class=”page-next” |
| Paginated load of filtered search | Only items from that category show The right number of products per page show The right product information should show | Automated Database Testing  Postman  Selenium |
| See product with multiple size options | See multiple size add to the cart buttons | Selenium.  Count the number of sizes available for a product that contains an indicator it has multiple sizes & assert that it is more than one. |
| See product with one option | See ‘Add to bag’ button | Selenium.  Click button with title=”add to bag”  Go to cart  Assert one item was added to the cart |
| Hover over product image | product image changes on hover | Selenium |
| List number of items | List number of items | Automated Database Testing  Postman |
| Sort by popularity | Products sorted by popularity | Automated Database Testing  Postman |
| Sort by price high to low | Products sorted by price high to low | Automated Database Testing  Postman |
| Sort by new in | Products sorted by new in | Automated Database Testing  Postman |
| Sort by price low to high | Products sorted by price low to high | Automated Database Testing  Postman |

## Post-purchase

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Automatic Refund through Admin - Success | Refund is processed Customer is notified on Success | Change status of product in test admin & check email |
| Automatic Refund through Admin - Failure (Card details have changed) | *Negative test*  Admin is notified on Failure. Admin to complete manually. Customer is notified on completion | Unit test  Check that admin receives email on failure of automatic refund |
| Customer cancels order | Order status is updated in order history page Ad email is sent to admin This starts the automatic refund process, if it began before the first cut-off date | Selenium test Change status of product in test admin & check email |

## Returns > Member

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Click <https://www.showpo.com/returns/>, having no eligible items | *Negative test*  Receive alert that customer has no order eligible for return if no orders have been made or they are outside of 30 days. | 3 API tests with Postman. Test data of  - items bought in the last 30 days that are therefore ineligible  - items bought more than 30 days ago that are in the eligible categories  - no orders made in lifetime of account or email address |
| Returns list of eligible items | Within the list are NOT cosmetics, lingerie, earrings, hair accessories, hats, hosiery, and items bought during final sale | API test with Postman |
| Click on eligible item | A form is presented | Selenium  Assert URL for the returns form |
| Click next step after form is presented | Emailed a return slip after 30 minutes. | Check email |
| Return successfully received by ShowPo | Store credit given to customer | Log in to account  Store credit has dollar amount that matches amount paid for item (discounted items are refunded discounted price) |

## Returns > Guest

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Submit email address in first stage of returns form | Go to next screen | Selenium  Enter email address in <https://www.showpo.com/returns/>  Click Continue button |
| Submit full 10 character order number in second stage of returns form | Taken to the next page to create a password Checks that the email address given made the order with the order number given | Selenium  Enter order number in  Click Continue button  Assert message ‘in order to process your return, we need you to have an account’ is on screen of next page |
| Submit less than 9 characters for order number in second stage of returns form | *Negative test - Boundary check*  An error message | Selenium  Enter 7 characters  Click ‘continue’ |
| Submit more than 9 characters for order number in second stage of returns form | *Negative test - Boundary check*  An error message | Selenium  Enter 11 characters  Click ‘continue’ |
| Submit zero characters for order number in second stage of returns form | *Negative test - Boundary check*  An error message | Selenium  Enter nothing  Click ‘continue’ |
| Create a password for the email address | Password is created | Selenium  Enter a random bunch of characters for password  Click ‘continue’ |
| Click ‘continue’ without entering email address or order number | *Negative test*  An alert with ‘Please fill in this field’. | Selenium  Enter nothing  Click ‘continue’ |

## Shipping

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| View price for same-day delivery for postcode | It matches what is in the document provided / the API given by the courier company | Postman |
| Orders over $50  Delivery within 5 business days | Free shipping | Unit test with mocking or DB query  Assert shipping cost option for ‘express’ says FREE |
| View shipping times for Australia | View shipping times for Australia | Go to <https://www.showpo.com/shipping-info/>  Select country on dropdown  Assert postcode field appears  Click “View shipping times” |

## Search

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Search by price in stock | Shows only products in price or price range | API with Postman |
| Search by brand in stock | Shows only products from brand | API with Postman |
| Search by size in stock | Shows only products categorized in size | API with Postman |
| Paginate after filter and sort | Filter and sort settings remain the same and there are no duplicates | API , Postman |
| Expected ‘out of stock’ behaviour | Hide or add notice + add to cart is disabled. *Pre-order* items that are out of stock. | API with Postman |
| Results appear | Where there are 50 or less items, only 1 page appears Where there are more than 50 , the option to click to page 2 appears. | Search a deep category and niche category, and count the number of products |
| Filter and sort items | Item count remains intact | API with Postman |
| Search with <https://github.com/minimaxir/big-list-of-naughty-strings/blob/master/blns.json> | *Negative test*  Failure with alert message | API with Postman. |

## Customer service and legal requirements

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Click on FAQ Accordion field to view information | FAQ Accordion displays text from relevant field when opened | Selenium |
| Click ‘ok’ on the cookies popup | It disappears | Selenium |
| Click FAQ link in footer | Goes to FAQ page | Selenium |
| Click T&CS link in footer | Go to T&CS PAGE | Selenium |
| Cookie policy follows GDPR | Cookie policy follows GDPR | Unit test |

## Help centre

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| Search help centre for words that have articles |  | Selenium  Write down how many articles one would expect for a search term like ‘faulty’. Search and see if it matches. |
| Search help centre for words that don’t have articles | *Empty state* | Selenium or unit test  Search for a term that wouldn’t have articles, like ‘cat’ |

## SEO

|  |  |  |
| --- | --- | --- |
| **Description of activity** | **Expected outcome** | **How to test** |
| All page loads | Breadcrumbs  Titles  Meta descriptions  FB schema  Twitter schema  Brand Structured Data | Selenium  Read response body |
| Product page loads | Breadcrumbs  Titles  Meta descriptions  FB schema  Twitter schema  Brand Structured Data  + Product Structured Data | Selenium  Read response body |