SauceDemo Test Execution Documentation

1 Successful Login

Test Name:
`test_successful_login`

Why we execute it:
To validate that a valid standard user can log in successfully.

Expected Outcome:
User is redirected to the Inventory page.

Command to execute:

```
pytest tests/auth/test_login.py::test_successful_login
```

2 Invalid Login

Test Name:
`test_invalid_login`

Why we execute it:
To ensure incorrect credentials show a proper error message.

Expected Outcome:
Error message is displayed on login page.

Command to execute:

```
pytest tests/auth/test_login.py::test_invalid_login
```

3 Locked User Login

Test Name:
`test_locked_user_login`

Why we execute it:
To verify that a locked user cannot access the application.

Expected Outcome:
Locked user error message is shown.

Command to execute:

```
pytest tests/auth/test_login.py::test_locked_user_login
```

4 Session Persistence After Refresh

Test Name:
`test_session_persistence_after_refresh`

Why we execute it:
To ensure user session remains active after refreshing the page.

Expected Outcome:
User remains on Inventory page after refresh.

Command to execute:

```
pytest
tests/auth/test_session_persistence.py::test_session_persistence_after_refre
sh
```

5 Session Persistence via Direct URL

Test Name:
`test_session_persistence_direct_url`

Why we execute it:
To ensure authenticated users can directly access inventory URL without being logged out.

Expected Outcome:
User remains authenticated and inventory page loads successfully.

Command to execute:

```
pytest
tests/auth/test_session_persistence.py::test_session_persistence_direct_url
```

6 Add to Cart

Test Name:
`test_add_to_cart`

Why we execute it:
To verify that products can be added to the shopping cart.

Expected Outcome:
Cart badge updates correctly.

Command to execute:

```
pytest tests/ecommerce/test_inventory_flow.py::test_add_to_cart
```

---

7 Remove from Cart

Test Name:
`test_remove_from_cart`

Why we execute it:
To ensure products can be removed from cart.

Expected Outcome:
Cart badge is removed or updated accordingly.

Command to execute:

```
pytest tests/ecommerce/test_inventory_flow.py::test_remove_from_cart
```

---

8 Complete Checkout Flow

Test Name:
`test_complete_checkout_flow`

Why we execute it:
To validate the complete purchase workflow from cart to order confirmation.

Expected Outcome:
Order confirmation page is displayed successfully.

Command to execute:

```
pytest tests/ecommerce/test_inventory_flow.py::test_complete_checkout_flow
```

---

9 Product Details Navigation

Test Name:
`test_product_details_navigation`

Why we execute it:
To ensure product detail page loads correctly when a product is clicked.

Expected Outcome:
Correct product details are displayed.

Command to execute:

```
pytest tests/products/test_product_details.py
```

---

10 Product Sorting A → Z

Test Name:
`test_sorting_name_ascending`

Why we execute it:
To verify products sort alphabetically from A to Z.

Expected Outcome:
Products appear in ascending alphabetical order.

Command to execute:

```
pytest tests/products/test_sorting.py::test_sorting_name_ascending
```

---

11 Product Sorting Z → A

Test Name:
`test_sorting_name_descending`

Why we execute it:
To verify products sort alphabetically from Z to A.

Expected Outcome:
Products appear in descending alphabetical order.

Command to execute:

```
pytest tests/products/test_sorting.py::test_sorting_name_descending
```

---

12 Price Sorting Low → High

Test Name:
`test_sorting_price_low_to_high`

Why we execute it:
To verify price sorting from lowest to highest.

Expected Outcome:
Products sorted in ascending price order.

Command to execute:

```
pytest tests/products/test_sorting.py::test_sorting_price_low_to_high
```

---

13 Price Sorting High → Low

Test Name:
`test_sorting_price_high_to_low`

Why we execute it:
To verify price sorting from highest to lowest.

Expected Outcome:
Products sorted in descending price order.

Command to execute:

```
pytest tests/products/test_sorting.py::test_sorting_price_high_to_low
```

---

14 API Mocking Test

Test Name:
`test_inventory_api_mock`

Why we execute it:
To validate application behavior when backend response is mocked.

Expected Outcome:
Application behaves according to mocked inventory data.

Command to execute:

```
pytest -m api
```

---

15 Visual Regression Test

Test Name:
`test_inventory_visual_snapshot`

Why we execute it:
To detect unintended UI changes in inventory page.

Expected Outcome:
Test passes if UI matches baseline snapshot.

Command to execute:

```
pytest -m visual
```

Test Categorization – SauceDemo Automation Framework

---

1 Smoke Test Suite

Purpose:
Validate critical user journey to ensure application is stable for further testing.

Command to execute:

```
pytest -m smoke
```

Included Tests

✅ Successful Login

- Ensures valid user can access application.

✅ Add to Cart

- Ensures product can be added successfully.

✅ Complete Checkout Flow

- Ensures end-to-end purchase works.

✅ Session Persistence (Refresh)

- Ensures user session remains active.

---

2 Regression Test Suite

Purpose:
Validate full functionality of application after changes.

Command to execute:

```
pytest -m regression
```

Included Tests

Authentication

- Successful Login
- Invalid Login
- Locked User Login
- Session Persistence (Refresh)
- Session Persistence (Direct URL)

Product Functionality

- Product Details Navigation
- Sorting A → Z
- Sorting Z → A
- Price Low → High
- Price High → Low

Ecommerce Flow

- Add to Cart
- Remove from Cart
- Complete Checkout

---

3 API Test Suite

Purpose:
Validate frontend behavior using mocked backend responses.

Command to execute:

```
pytest -m api
```

Included Test

- Inventory API Mocking

---

4 Visual Regression Suite

Purpose:
Detect unintended UI changes.

Command to execute:

```
pytest -m visual
```

Included Test

- Inventory Page Visual Snapshot

---

5 Full Suite Execution

Purpose:
Execute entire automation coverage.

Command:

```
pytest
```

---

6 Parallel Execution (Full Regression)

Purpose:
Speed up execution for larger test suites.

Command:

```
pytest -m regression -n auto
```

Execution Strategy Summary

Suite	Purpose	When to Run	Command
Smoke	Validate core stability	Every deployment	`pytest -m smoke`
Regression	Validate full functionality	Release validation	`pytest -m regression`
API	Validate backend handling	API changes	`pytest -m api`
Visual	Detect UI regressions	UI updates	`pytest -m visual`
Full Suite	Complete validation	Major release	`pytest`
