Test Plan — Part 1 (UI: https://www.saucedemo.com)

Main features

1. Login
2. Cart + Checkout (basic purchase flow)

Scope & risks (summary)

- Environment: public, volatile data; tests should be short.
- Risks: invalid credentials/lock; flakiness due to animations; network dependency.

Proposed scenarios

A) Login

ID	Scenario	Expected result
UI-01	Check login valid user	should log in successfully
UI-02	Check login invalid user	it should not be possible to log in
UI-03	Check login locked user	it should not be possible to log in with a blocked user

B) Cart + checkout

ID	Scenario	Expected result
UI-01	Check the entire purchase flow	The purchase must be made successfully
UI-02	Check required field in checkout step	It should highlight the lack of completion of a mandatory field
UI-03	Remove item from cart	item must be removed from cart successfully

Architecture (UI)

- **Pattern**: Actions (high-level methods that orchestrate steps) + Elements (centralized locators). This avoids heavy POM coupling and makes the tests readable.
- **Data: users.json** (credentials) and light generation with Faker when useful.
- **Best practices:** stable locators (role, name, data-test), explicit asserts, Playwright default retry, test.step for narration.

Test Plan — Part 2 (API: https://petstore.swagger.io)

Selected APIs

1. **POST /pet** (Create pet)
2. **GET /pet/{petId}** + **DELETE /pet/{petId}** (get and remove)
3. **GET pet/{petId}/invalidpetId** (Check the response with a invalid petId)

Proposed scenarios

A) POST/GET/DELETE pet

ID	Scenario	Expected result
API-01	{POST} Create a new pet	return 200; check response time; check a valid body
API-02	{GET} Get pet by petId	return 200; check response time; check a valid body
API-03	{DELETE} Delete petId	Return 200; Delete a petId
API-03	{GET} Invalid petId	return 404 or 400; check response time; check a valid body with invalid data