# MASFIQUR RAHMAN

Senior QA Work Sample | Playwright Automation + AI-Assisted QA

Prepared for: **Senior QA Engineer - B2C / SaaS** (Manual + Automation)
Focus: Playwright | API (Postman) | Cursor / Copilot workflows | Web - Mobile - Desktop
Independent freelancer - not an agency

Resume PDF: masfiqur.com/cv/MD-Masfiqur-Rahman-CV.pdf | Portfolio: www.masfiqur.com

| 14+ | Playwright | Cursor | Multi-platform |
|---|---|---|---|
| Years QA / B2C & SaaS | Primary automation stack | AI-assisted test drafting | Web - mobile - desktop |

## About this sample

Illustrative artifacts showing how I own QA for B2C/SaaS products: risk-based plans, maintainable Playwright automation, API checks, and AI-assisted drafting with human adversarial review. Anonymized demos - not client-confidential materials.

## How I own QA across platforms

- Coordinate with PMs/devs on goals ? define coverage for core, regression, and edges
- Manual + exploratory first on new features; lock winners into automation
- Optimize suite for ROI: P0 flows, flaky-resistant locators, failure-path coverage
- Validate performance/stability signals and security-minded cases before release
- Clear English defects: steps, expected vs actual, severity, evidence, verify-fix bar

## Contents



*Tip: pages 2-3 are the automation proof the job asks for; page 4 shows planning & defects.*

## 1. Playwright Automation Examples

I write stable, maintainable Playwright tests: role/label locators, expect()-based waits, and explicit failure paths - not only happy-path smoke.

### Example A - Checkout recovers after payment API failure (B2C critical path)

```
import { test, expect } from '@playwright/test';

test('checkout recovers after payment API 503', async ({ page }) => {
  await page.goto('/cart');
  await page.getByRole('button', { name: 'Checkout' }).click();

  await page.route('**/api/payments/charge', async (route) => {
    await route.fulfill({ status: 503, body: JSON.stringify({ error: 'unavailable' }) });
  });

  await page.getByLabel('Card number').fill('4242424242424242');
  await page.getByRole('button', { name: 'Pay now' }).click();

  await expect(page.getByText(/payment failed|try again/i)).toBeVisible();
  await expect(page.getByRole('button', { name: 'Pay now' })).toBeEnabled();
  await expect(page.getByText(/processing/i)).toHaveCount(0);
  // Cart preserved - user can retry without losing items
  await expect(page.getByTestId('cart-item-count')).not.toHaveText('0');
});
```

### Example B - SaaS auth session: login ? gated feature ? logout clears access

```
test('session gates billing and clears on logout', async ({ page }) => {
  await page.goto('/login');
  await page.getByLabel('Email').fill(process.env.QA_USER!);
  await page.getByLabel('Password').fill(process.env.QA_PASS!);
  await page.getByRole('button', { name: 'Sign in' }).click();
  await expect(page.getByRole('heading', { name: 'Dashboard' })).toBeVisible();

  await page.goto('/billing');
  await expect(page.getByText(/current plan/i)).toBeVisible();

  await page.getByRole('button', { name: 'Log out' }).click();
  await page.goto('/billing');
  await expect(page).toHaveURL(/login/);
  await expect(page.getByText(/current plan/i)).toHaveCount(0);
});
```

*Also comfortable maintaining Cypress and Selenium suites when a codebase already uses them - I optimize for clean structure and low flake, not tool dogma.*

## 2. Cursor / Copilot in My QA Workflow

Your post asks how AI tools accelerate test development. Here is my real loop - speed with accountability.

| Step | What I ask Cursor / Copilot | What I still own |
|---|---|---|
| 1. Draft | Scaffold Playwright spec from acceptance criteria / PR summary | Scope: which flows are P0 vs nice-to-have |
| 2. Locators | Suggest getByRole / getByLabel over brittle CSS | Reject fragile selectors; align with a11y-friendly UI |
| 3. Edges | Propose timeout, empty, 4xx/5xx, permission cases | Pick adversarial cases that match real user risk |
| 4. API | Draft Postman/REST assertions from OpenAPI snippets | Validate against staging; catch contract drift |
| 5. Review | Summarize PR risk areas for exploratory focus | Read the diff myself - never trust "regenerate until green" |

### When I do NOT trust AI output

- Assertions that only echo mocks (tautological tests)
- Happy-path-only suites with no failure/recovery coverage
- Security, payments, or auth changes without manual adversarial pass
- Flake "fixes" that hide race conditions with arbitrary sleeps

*AI amplifies engineering I already understand - it does not replace judgment, staging proof, or clear defect communication.*

## 3. API Testing + AWS / GCP Environments

UI green is incomplete without API and environment proof - especially for SaaS and B2C flows that hit billing, auth, and async jobs.

### Postman / REST checks I routinely include

- Auth: token issue/refresh/expiry; unauthorized vs forbidden distinction
- Contract: required fields, status codes, error shape consistency
- Idempotency: double-submit checkout / retry-safe POSTs
- Data: create ? read ? update ? soft-delete consistency across environments
- Negative: malformed payloads, rate limits, missing headers

### Playwright API request smoke (paired with UI)

```
test('billing API returns plan for authenticated user', async ({ request }) => {
  const login = await request.post('/api/auth/login', {
    data: { email: process.env.QA_USER, password: process.env.QA_PASS },
  });
  expect(login.ok()).toBeTruthy();
  const { token } = await login.json();

  const billing = await request.get('/api/billing/plan', {
    headers: { Authorization: `Bearer ${token}` },
  });
  expect(billing.status()).toBe(200);
  const body = await billing.json();
  expect(body).toMatchObject({ plan: expect.any(String), status: 'active' });
});
```

### Cloud / environment discipline

| Concern | What I verify | Why it matters |
|---|---|---|
| Config drift | Feature flags / env vars match intended stage (AWS/GCP) | Staging green ? prod-safe |
| Secrets | No real keys in tests; use stage credentials only | Security + audit hygiene |
| Perf smoke | P95 of critical API under light load; error rate under stress | Catch obvious scale risks early |
| Stability | Retry/backoff UX + job completion vs UI badge | Silent stalls erode trust |

## 4. Multi-Platform Coverage Matrix (Excerpt)

I plan coverage once, then execute where risk lives - web, mobile, and desktop - instead of duplicating every case on every surface.

| Area | Web | Mobile | Desktop | Automation note |
|---|---|---|---|---|
| Auth / session | P0 | P0 | P0 | Playwright web; device smoke manual |
| Core purchase / upgrade | P0 | P0 | P1 | E2E + API assert on charge result |
| Settings / profile | P1 | P1 | P1 | Smoke + visual sanity |
| Offline / flaky net | P2 | P0 | P2 | Manual + route.abort() cases |
| Notifications | P1 | P0 | P1 | Permission deny + deep link |
| Accessibility basics | P1 | P2 | P1 | Keyboard + role locators |

### Risk-based test plan snippet (new feature)

| | |
|---|---|
| **Feature** | Self-serve plan upgrade (SaaS) |
| **Goal** | Paid plan unlocks features; failed charge never unlocks |
| **P0 cases** | Happy upgrade - decline card - refresh mid-pay - double-click Pay |
| **Exploratory** | Mobile Safari wallet UX - slow 3G - logged-out deep link to /upgrade |
| **Exit** | P0 automated or checklist-green; Critical/High fixed or waived with owner |

## 5. Sample Defect + Strengths Alignment

### BUG-SAAS-114 - Success toast after failed subscription charge (unlocks gated UI temporarily)

| | |
|---|---|
| **Severity** | **Critical** |
| **Platforms** | Web (Chrome) - iOS Safari - Staging on AWS |
| **Steps** | Free user ? Upgrade ? decline test card ? Confirm ? open gated feature ? refresh |
| **Expected** | Clear failure; plan stays Free; gated features remain locked |
| **Actual** | "Subscription activated" toast; temporary unlock; billing still Free after refresh |
| **Evidence** | Screenshots + 45s Loom; network shows 402 while UI shows success |
| **Verify-fix** | Same card decline ? error UI; no unlock; Playwright Example A stays green |

### Mapped to your necessary strengths

| Strength | How I demonstrate it |
|---|---|
| Organization & Planning | Risk matrices, P0/P1 coverage, entry/exit criteria before each release |
| Proactivity | I hunt failure paths and process gaps - I don't wait for a ticket list |
| Attention to Detail | UI/API mismatches, session leaks, temporary unlocks, wording that misleads users |
| Communication | Concise English reports with repro, evidence, severity, and a clear verify-fix bar |

### Closing

I am an independent freelancer ready to own manual + automated QA for your B2C/SaaS stack - Playwright-first automation, Cursor-accelerated drafting with human review, and clear communication from first bug to verified fix.

---

Masfiqur Rahman | Senior QA Engineer | 14+ years | Independent
Playwright - Cypress - Selenium - Postman - Cursor - Web / Mobile / Desktop
Resume: masfiqur.com/cv/MD-Masfiqur-Rahman-CV.pdf | www.masfiqur.com