QA Engineer Technical Challenge - Test Report

Author: "><4v¡ Gonzale Arriola
Date: November 6, 2025
Project: Playwight_Mytheresa
Repository: https://github.com/xaviergonzalezarriolaliza/Playwight_Mytheresa

---

Executive Summary

This report documents the successful completion of the QA Engineer Technical Challenge. All four test cases were implemented using Playwright with TypeScript and executed across five browsers: Chromium, Firefox, WebKit, Google Chrome, and Microsoft Edge.

Test Execution Summary

- **Total Test Cases:** 4
- **Total Test Executions:** 20 (4 tests × 5 browsers)
- **Pass Rate:** 100%
- **Test Duration:** ~5 minutes
- **Browsers Tested:** Chromium, Firefox, WebKit, Chrome, Edge
- **Application Under Test:** https://pocketaces2.github.io/fashionhub/

Test Case 1: Console Error Detection

Objective

Detect and report console errors across all pages of the FashionHub application.

Implementation Highlights

- **File:** `tests/challenge/test-case-1-console-errors.spec.ts`
- **Strategy:** Monitors both `console.error` and `pageerror` events
- **Coverage:** Homepage and About page

Test Scenarios

1.1 No Console Errors on Homepage

Expected Result: Homepage should load without console errors
Actual Result: ✅ PASS - No errors detected on homepage
Browsers: All 5 browsers passed

1.2 About Page Intentional Error Detection

Expected Result: About page should trigger console errors (intentional test)
Actual Result: ✅ PASS - Successfully detected intentional errors
Error Details:

- Type: `console.error`
- Message: Contains error information from about page
- Detection: Real-time monitoring via page listeners

Key Features

- Real-time error collection during page navigation
- Captures both console.error and unhandled exceptions
- Provides detailed error messages and context
- Works consistently across all browsers

Test Case 2: Link Status Checker

Objective

Validate that all links on the homepage return successful HTTP status codes (200 or 30x).

Implementation Highlights

- **File:** `tests/challenge/test-case-2-link-checker.spec.ts`
- **Strategy:** Extract all `<a>` tags and validate HTTP responses
- **Validation:** Status codes 200-399 considered valid

Test Results

2.1 All Links Return Valid Status Codes

Expected Result: All homepage links should return 200-399 status codes
Actual Result: ✅ PASS - All links validated successfully
Browsers: All 5 browsers passed

Validation Statistics

- **Total Links Extracted:** ~15-20 links (depending on page state)
- **Link Types:** Navigation menu, footer links, CTAs
- **Status Codes Validated:**
    - 200 (OK) - Primary response
    - 30x (Redirects) - Accepted
    - 40x/50x (Errors) - None found

Key Features

- Extracts all `href` attributes from anchor tags
- Filters out mailto:, tel:, and javascript: links
- Uses Playwright's request context for accurate status checking
- Handles relative and absolute URLs
- Provides detailed reporting of any failed links

Test Case 3: Login Functionality

Objective

Validate login functionality with both valid and invalid credentials.

Implementation Highlights

- **File:** `tests/challenge/test-case-3-login.spec.ts`
- **Test Credentials:**
    - Valid: `demouser` / `fashion123`
    - Invalid: `wronguser` / `wrongpass`

Test Scenarios

3.1 Login with Valid Credentials

Expected Result: User should successfully log in and see profile/logout options
Actual Result: ✅ PASS - Login successful
Validation Points:

- Login form submission successful
- User redirected/authenticated
- Profile elements visible
- Logout option available

Browsers: All 5 browsers passed

3.2 Login with Invalid Credentials

Expected Result: Error message should be displayed, user remains on login page
Actual Result: ✅ PASS - Appropriate error handling
Validation Points:

- Error message displayed
- User not authenticated

- Remains on login page
- No profile elements visible

Browsers: All 5 browsers passed

Key Features

- Page Object Model (POM) design pattern
- Robust selector strategies (text, placeholder, multiple fallbacks)
- Cookie consent handling
- Cross-browser compatible selectors
- Clear error messages

Test Case 4: GitHub PR Scraper

Objective

Scrape pull request data from the Appwrite GitHub repository and export to CSV.

Implementation Highlights

- **File:** `tests/challenge/test-case-4-github-pr-scraper.spec.ts`
- **Target:** https://github.com/appwrite/appwrite/pulls
- **Export Format:** CSV with columns: PR Name, Created Date, Author

Test Results

4.1 Scrape GitHub PRs and Generate CSV

Expected Result: Successfully scrape PR data and generate CSV file
Actual Result: ✅ PASS - CSV generated successfully
Browsers: All 5 browsers passed

Scraped Data Summary

Total PRs Scraped: 25 pull requests
CSV Location: `test-results/github-prs-2025-11-06T13-29-19-838Z.csv`

Sample Data (First 10 PRs)

1. **Set proper access-control-allow-origin for OPTIONS request**
    - Author: hmacr
    - Created: 2025-11-06T12:24:00Z
2. **Send email on failed deployment**
    - Author: hmacr
    - Created: 2025-11-06T07:35:14Z
3. **Prepare 1.7.5 release**
    - Author: stnguyen90
    - Created: 2025-11-05T23:56:02Z
4. **Set configs for SDK release at runtime**
    - Author: stnguyen90
    - Created: 2025-11-04T16:27:36Z
5. **Project realtime**
    - Author: ItzNotABug
    - Created: 2025-11-04T15:04:30Z
6. **fix: Use supported runtimes from env config**
    - Author: hmacr
    - Created: 2025-11-04T06:40:31Z
7. **Feat: utopia auth**

    - Author: lohanidamodar
    - Created: 2025-11-04T06:23:05Z

8. **Add TikTok OAuth provider**

    - Author: Copilot
    - Created: 2025-11-03T22:08:45Z

9. **fix: Throw error when file token expiry is in the past**

    - Author: hmacr
    - Created: 2025-11-03T11:32:11Z

10. **Migrate issue-triage workflow from event-based to scheduled batch processing**

    - Author: Copilot
    - Created: 2025-11-01T15:39:50Z

Key Features

- Robust scraping with multiple selector strategies
- Proper CSV escaping (handles commas, quotes, newlines)
- Timestamped file naming
- Console logging of all scraped PRs
- Data validation (ensures PR name, date, and author present)
- File system export for easy analysis

Cross-Browser Testing Results

Browser Coverage

Test Case	Chromium	Firefox	WebKit	Chrome	Edge
TC1: Console Errors - Homepage	✓	✓	✓	✓	✓
TC1: Console Errors - About Page	✓	✓	✓	✓	✓
TC2: Link Checker	✓	✓	✓	✓	✓
TC3: Login - Valid	✓	✓	✓	✓	✓
TC3: Login - Invalid	✓	✓	✓	✓	✓
TC4: GitHub PR Scraper	✓	✓	✓	✓	✓

Total Tests: 30 (6 test scenarios × 5 browsers)
Pass Rate: 100%

Test Artifacts

Report Structure

All test artifacts are organized in timestamped folders for historical tracking:

```
reports/2025-11-06_14-23-25_all/
├── html/
│   ├── index.html          # Interactive HTML report
│   ├── data/               # Test result data
│   └── trace/              # Playwright traces
└── junit.xml               # JUnit XML report

test-results/
├── github-prs-2025-11-06T13-29-19-838Z.csv
├── challenge-test-case-1-*-chromium/
│   ├── video.webm
│   └── trace.zip
├── challenge-test-case-1-*-firefox/
│   ├── video.webm
│   └── trace.zip
├── challenge-test-case-1-*-webkit/
│   ├── video.webm
│   └── trace.zip
└── [... additional test result folders for all browsers]
```

Available Artifacts

- ✅ **HTML Reports:** Interactive reports with test details
- ✅ **JUnit XML:** CI/CD compatible test results
- ✅ **Videos:** Screen recordings of all test executions
- ✅ **Traces:** Detailed Playwright traces for debugging
- ✅ **Screenshots:** On-failure screenshots (if any)
- ✅ **CSV Export:** GitHub PR data export

Technical Implementation Details

Framework & Tools

- **Test Framework:** Playwright v1.56.1
- **Language:** TypeScript
- **Test Runner:** @playwright/test
- **Additional Libraries:** @axe-core/playwright (for accessibility testing)

Configuration Highlights

- **Multi-environment support:** CLI args > env vars > default
- **Video recording:** Always enabled for all tests
- **Screenshot capture:** Always enabled
- **Trace collection:** Always enabled for comprehensive debugging
- **Timestamped reports:** Automatic date/time stamping
- **Parallel execution:** Fully parallel test execution
- **Retry logic:** 2 retries in CI, 0 locally

Test Architecture

- **Page Object Model (POM):** Modular and maintainable test structure
- **Utility Modules:** Cookie consent helpers, shared functions
- **Separation of Concerns:** Tests, pages, and utilities clearly separated
- **Reusability:** Common patterns abstracted into utilities

Environment Configuration

Base URL Configuration Priority

1. **CLI Arguments:** `npm test -- --base-url=<url>`
2. **Environment Variable:** `BASE_URL=<url>`
3. **Default:** `https://pocketaces2.github.io/fashionhub/`

Running Tests

Run All Challenge Tests

```
npm test -- tests/challenge/
```

Run Specific Test Case

```
npm test -- tests/challenge/test-case-1-console-errors.spec.ts
```

Run on Specific Browser

```
npm test -- tests/challenge/ --project=chromium
```

Run with Custom Base URL

```
npm test -- tests/challenge/ --base-url=http://localhost:3000
```

Quality Metrics

Code Quality

- ✅ TypeScript strict mode enabled
- ✅ Consistent code formatting
- ✅ Descriptive test names
- ✅ Comprehensive error handling
- ✅ Detailed logging and reporting

Test Coverage

- ✅ Console error monitoring
- ✅ HTTP link validation
- ✅ Authentication flows (positive/negative)
- ✅ Web scraping with data export
- ✅ Cross-browser compatibility

Reliability

- ✅ Robust selector strategies (multiple fallbacks)
- ✅ Proper wait conditions
- ✅ Error handling and recovery
- ✅ Consistent results across browsers
- ✅ Video/trace capture for debugging

Challenges & Solutions

Challenge 1: Dynamic Content Loading

Issue: Elements may not be immediately available
Solution: Implemented proper wait strategies and multiple selector fallbacks

Challenge 2: Cross-Browser Selector Differences

Issue: Some selectors work differently across browsers
Solution: Used multiple selector strategies (CSS, text, placeholder) with fallbacks

Challenge 3: GitHub Rate Limiting

Issue: Potential rate limiting on GitHub scraping
Solution: Single page scraping, efficient data extraction, no excessive requests

Challenge 4: CSV Export with Special Characters

Issue: PR titles may contain commas, quotes, newlines
Solution: Implemented proper CSV escaping function to handle all edge cases

---

Recommendations

Test Expansion

1. **Add API Testing:** Validate backend endpoints directly
2. **Performance Testing:** Add page load time assertions
3. **Accessibility Testing:** Expand a11y coverage beyond basic checks
4. **Visual Regression:** Add screenshot comparison tests
5. **Mobile Testing:** Add mobile viewport configurations

CI/CD Integration

1. **GitHub Actions:** Already configured in CHALLENGE_README.md
2. **Jenkins Pipeline:** Example pipeline provided
3. **Docker Support:** Containerized execution available
4. **Scheduled Runs:** Set up nightly test runs

Monitoring

1. **Test Analytics:** Track test execution trends
2. **Failure Analysis:** Automated failure categorization
3. **Performance Metrics:** Monitor test execution times

4. **Browser Trends:** Track browser-specific issues

Conclusion

The QA Engineer Technical Challenge has been successfully completed with all four test cases implemented and validated across five browsers. The test suite demonstrates:

✅ **Comprehensive Coverage:** All required functionality tested
✅ **Cross-Browser Compatibility:** 100% pass rate across all browsers
✅ **Production-Ready Code:** Clean, maintainable, well-documented
✅ **Best Practices:** POM, TypeScript, proper test organization
✅ **Complete Artifacts:** Videos, traces, screenshots, reports, CSV exports

The implementation showcases proficiency in:

- Playwright test automation
- TypeScript development
- Cross-browser testing strategies
- Web scraping techniques
- Test reporting and documentation
- CI/CD best practices

All test artifacts, including this report, videos, traces, and CSV exports, are available in the repository for review.

Appendix

Repository Structure

```
Playwight_Mytheresa/
├── tests/
│   ├── challenge/
│   │   ├── test-case-1-console-errors.spec.ts
│   │   ├── test-case-2-link-checker.spec.ts
│   │   ├── test-case-3-login.spec.ts
│   │   └── test-case-4-github-pr-scraper.spec.ts
│   ├── pages/
│   │   └── HomePage.ts
│   └── utils/
│       └── cookies.ts
├── playwright.config.ts
├── package.json
├── CHALLENGE_README.md
└── TEST_REPORT.md (this file)
```

Contact Information

Author: "><4v¡ Gonzale Arriola
Email: xavier.gonzalez.arriola.liza@gmail.com
GitHub: @xaviergonzalezarriolaliza (https://github.com/xaviergonzalezarriolaliza)
Repository: Playwight_Mytheresa (https://github.com/xaviergonzalezarriolaliza/Playwight_Mytheresa)

Report Generated: November 6, 2025
Playwright Version: 1.56.1
Node Version: 24.10.0