SECTIONS 10-13: Best Practices, Roles, Mistakes & Workflow Summary

SECTION 10 — Enterprise Best Practices

10.1 Code Quality Standards

Coding Standards Checklist:

Area	Standard	Example
Locators	Use data-testid or stable selectors	`[data-testid="login-btn"]`
Waits	Never use hardcoded waits	Use `waitFor()` instead
Assertions	Use specific assertions	`toHaveURL()` not just `toBeTruthy()`
Page Objects	One class per page	`LoginPage.ts` , `HomePage.ts`
Test Data	External files, not hardcoded	`test-data/loginData.ts`
Naming	Descriptive test names	`should show error for empty password`

Code Review Checklist:

```
□ No hardcoded values (URLs, credentials)
□ Proper error handling
□ Descriptive variable names
□ Page Object pattern followed
□ No unnecessary waits
□ Assertions are specific
□ Test covers stated scenario
□ No console.log statements
□ Follows naming conventions
□ Test tags applied (@smoke, @regression)
```

10.2 Mandatory PR Reviews

PR Requirements:

Requirement	Description
1+ Approval	At least one reviewer must approve
Tests Pass	All CI checks must be green
No Conflicts	Must be mergeable
Updated Branch	Must include latest develop changes

Review Focus Areas:

```
REVIEWER CHECKLIST:

□ Test Logic
  └ Does test actually verify what it claims?

□ Locators
  └ Are they stable and maintainable?

□ Test Data
  └ Is sensitive data handled via secrets?

□ Error Handling
  └ Will test fail clearly if issue occurs?

□ Readability
  └ Can another engineer understand this?

□ Performance
  └ No unnecessary waits or loops?
```

10.3 Linting

ESLint Configuration:

```json
// .eslintrc.json
{
    "extends": [
        "eslint:recommended",
        "plugin:@typescript-eslint/recommended",
        "plugin:playwright/recommended"
    ],
    "rules": {
        "@typescript-eslint/no-unused-vars": "error",
        "playwright/no-wait-for-timeout": "error",
        "playwright/prefer-web-first-assertions": "error",
        "no-console": "warn"
    }
}
```

Run Linting in Pipeline:

```yaml
- name: Lint code
  run: npm run lint

- name: Type check
  run: npx tsc --noEmit
```

10.4 Test Tagging

Tag Categories:

Tag	Purpose	Example
`@smoke`	Quick sanity checks	Login, homepage load
`@regression`	Full test suite	All test cases
`@critical`	Business-critical paths	Payment, checkout
`@slow`	Long-running tests	Reports, uploads
`@flaky`	Known unstable tests	Investigate later

Applying Tags:

```
test('verify login @smoke @login', async ({ page }) => {
    // Critical path test
});

test('complex workflow @regression @slow', async ({ page }) => {
    // Long running test
});
```

Running by Tag:

```
npx playwright test --grep @smoke        # Smoke tests only
npx playwright test --grep-invert @slow  # Exclude slow tests
```

10.5 Retry Strategy

Configuration Levels:

```
// playwright.config.ts
export default defineConfig({
    // Global retries
    retries: process.env.CI ? 2 : 0,

    // Project-specific
    projects: [
        {
            name: 'chromium',
            retries: 3,    // More retries for Chrome
        },
    ],
});
```

Test-Level Retry:

```
test('potentially flaky test', async ({ page }) => {
    test.info().config.retries = 5;   // Extra retries
    // test code
});
```

10.6 Flaky Test Handling

What is a Flaky Test?

A test that sometimes passes and sometimes fails without code changes.

Identification:

```
Test Results Over 10 Runs:
Run 1: ✅ Pass
Run 2: ✅ Pass
Run 3: ❌ Fail    ← Flaky!
Run 4: ✅ Pass
Run 5: ✅ Pass
```

Common Causes & Solutions:

Cause	Solution
Race conditions	Add proper waits (`waitFor`)
Network timing	Use `networkidle` state
Animation	Wait for animation completion
Test data pollution	Isolate test data
Parallel conflicts	Make tests independent

Handling Strategy:

```
FLAKY TEST WORKFLOW:

1. Identify
   └ Monitor test results for inconsistency

2. Quarantine
   └ Add @flaky tag
   └ Exclude from critical pipelines

3. Investigate
   └ Run repeatedly: npx playwright test --repeat-each=10
   └ Add traces: trace: 'on'

4. Fix
   └ Apply proper fixes
   └ Remove @flaky tag

5. Monitor
   └ Watch for recurrence
```

10.7 Documentation Standards

What to Document:

Item	Location	Content
Setup guide	README.md	How to start
Test guide	docs/TESTING.md	How to run tests
Framework guide	docs/FRAMEWORK.md	Architecture
Change log	CHANGELOG.md	What changed

Test File Headers:

```
/**
 * Login Page Test Suite
 *
 * Tests the login functionality including:
 * - Valid credential login
 * - Invalid credential handling
 * - Empty field validation
 * - Remember me feature
 *
 * @module login
 * @tags @smoke @regression
 */
```

10.8 Security Practices

Security Checklist:

Practice	Implementation
No hardcoded secrets	Use GitHub Secrets
API keys	Store in secrets
Test data	Use fake data generators
Screenshots	Don't capture sensitive data
Logs	Mask sensitive information

Secure Secret Usage:

```
// BAD ❌
const password = 'myP@ssw0rd123';

// GOOD ✅
const password = process.env.TEST_PASSWORD;
```

10.9 Audit & Compliance

Audit Trail Maintenance:

```
MAINTAIN HISTORY OF:

□ All test runs (CI/CD keeps these)
□ Code changes (Git keeps these)
□ PR approvals (GitHub keeps these)
□ Deployment records
□ Access changes
```

Compliance Requirements:

Requirement	Implementation
Traceability	Link tests to requirements
Evidence	Store test reports
Access Control	Restrict sensitive access
Change Approval	Require PR reviews

SECTION 11 — Roles & Responsibilities

11.1 Junior Automation Engineer

Experience Level: 0-2 years

Day-to-Day Activities:

```
Morning:
├── Pull latest code
├── Check assigned tickets
└── Review any feedback on PRs

During Day:
├── Write new test cases
├── Fix assigned bugs
├── Run tests locally
└── Ask questions when stuck

End of Day:
├── Push completed work
├── Update ticket status
└── Note blockers
```

Key Responsibilities:

Responsibility	Priority
Write new tests	High
Fix failing tests	High
Follow standards	High
Learn framework	Medium
Document work	Medium

Skills to Develop:

- Playwright fundamentals
- TypeScript basics
- Git workflow
- Page Object Model
- Debugging skills

11.2 Senior Automation Engineer

Experience Level: 2-5 years

Responsibilities:

Area	Tasks
Technical	Design complex tests, handle edge cases
Quality	Review PRs, enforce standards
Mentoring	Guide junior engineers
Troubleshooting	Debug flaky/complex failures

Decision-Making Authority:

```
CAN DECIDE:
├── Test implementation approach
├── Which locators to use
├── Test data structure
└── Code review outcomes

SHOULD ESCALATE:
├── Framework architecture changes
├── New tool adoption
├── Major refactoring
└── Test coverage gaps
```

11.3 Automation Lead

Experience Level: 5+ years

Strategic Responsibilities:

Area	Focus
Planning	Test strategy, sprint planning
Coordination	Work with dev/QA teams
Reporting	Metrics, coverage reports
Technical	Architecture decisions
Team	Hiring, training, reviews

Metrics Owned:

```
AUTOMATION LEAD DASHBOARD:

Test Coverage:      ████████░░ 80%
Pass Rate:          █████████░ 92%
Execution Time:     35 minutes
Flaky Rate:         ██░░░░░░░░ 5%
Open Defects:       12
Sprint Velocity:    15 tests/sprint
```

11.4 DevOps / CI Owner

Focus Area: Infrastructure & Pipelines

Responsibilities:

Area	Tasks
CI/CD	Setup and maintain pipelines
Infrastructure	Manage runners, environments
Secrets	Handle credentials securely
Monitoring	Pipeline health, alerts
Optimization	Reduce execution time

Technical Ownership:

```
OWNS:
├── .github/workflows/
├── GitHub Actions configuration
├── Secret management
├── Environment setup
├── Runner management
└── Pipeline optimization
```

11.5 QA Manager

Strategic Focus:

Area	Responsibility
Strategy	Overall quality approach
Resources	Team allocation, hiring
Budget	Tools, infrastructure costs
Stakeholders	Progress reporting
Process	Methodology improvements

Reporting Expectations:

```
MONTHLY REPORT TEMPLATE:

Executive Summary
├── Overall test health
├── Coverage improvements
├── Key risks

Metrics
├── Test execution trends
├── Defect trends
├── Automation ROI

Roadmap
├── Next month priorities
├── Resource needs
└── Risk mitigation
```

SECTION 12 — Common Enterprise Mistakes & Solutions

12.1 Merge Conflicts

Common Scenarios:

Scenario	Cause	Prevention
Same file edited	Two people worked on same file	Communicate, smaller files
Outdated branch	Didn't pull before pushing	Always pull before work
Long-lived branches	Branch existed too long	Merge frequently

Resolution Steps:

```
# 1. Pull latest develop
git checkout develop
git pull origin develop

# 2. Go back to your branch
git checkout feature/your-branch

# 3. Merge develop into your branch
git merge develop

# 4. If conflicts appear, open conflicted files
# 5. Resolve conflicts manually (remove markers)
# 6. Stage resolved files
git add .

# 7. Complete the merge
git commit -m "fix: resolve merge conflicts with develop"

# 8. Push
git push origin feature/your-branch
```

12.2 Broken Pipelines

Troubleshooting Table:

Error	Likely Cause	Solution
"npm ci failed"	Missing package-lock.json	Commit package-lock.json
"Browser not found"	Missing install step	Add `playwright install --with-deps`
"Timeout exceeded"	Tests too slow or stuck	Increase timeout, fix stuck test

Error	Likely Cause	Solution
"Out of memory"	Too many parallel workers	Reduce worker count
"Permission denied"	File permission issue	Check file permissions
"Secret not found"	Secret not configured	Add secret in settings

Debugging Pipeline:

```yaml
# Add debug output
- name: Debug info
  run: |
    echo "Node version: $(node --version)"
    echo "NPM version: $(npm --version)"
    echo "Current directory: $(pwd)"
    ls -la
```

12.3 Unstable Tests

Stability Checklist:

Issue	Check	Fix
Timing issues	Are you using hardcoded waits?	Use `waitFor()`
Selector problems	Is selector specific enough?	Use data-testid
Test isolation	Does test depend on others?	Make independent
Data issues	Is data shared between tests?	Isolate test data
Race conditions	Are you waiting for async ops?	Add proper waits

Making Tests Stable:

```
// BAD ❌ - Hardcoded wait
await page.waitForTimeout(3000);

// GOOD ✅ - Wait for element
await page.locator('#result').waitFor({ state: 'visible' });

// BAD ❌ - Flaky selector
await page.locator('.btn').click();

// GOOD ✅ - Stable selector
await page.locator('[data-testid="submit-btn"]').click();
```

12.4 Hardcoded Data

Problem:

```
// BAD ❌
const username = 'testuser123';
const password = 'P@ssw0rd!';
const apiUrl = 'https://api.prod.example.com';
```

Solution:

```
// GOOD ✅ - Use environment variables
const username = process.env.TEST_USERNAME;
const password = process.env.TEST_PASSWORD;
const apiUrl = process.env.API_URL;

// GOOD ✅ - Use test data files
import { loginData } from '../test-data/loginData';
await loginPage.login(loginData.username, loginData.password);
```

12.5 Poor Branching

Anti-Patterns:

❌ Bad Practice	✅ Best Practice
Commit to main directly	Use feature branches

❌ Bad Practice	✅ Best Practice
Never merge develop	Merge frequently
Huge feature branches	Small, focused branches
Vague branch names	Descriptive names
Never delete branches	Clean up after merge

Branch Hygiene:

```
# List all branches
git branch -a

# Delete merged local branches
git branch --merged | grep -v "main\|develop" | xargs git branch -d

# Delete remote merged branch
git push origin --delete feature/old-branch
```

SECTION 13 — Real-World Enterprise Workflow

Summary

13.1 End-to-End Flow

```
┌─────────────────────────────────────────────────────────────────┐
│               COMPLETE ENTERPRISE WORKFLOW                       │
├─────────────────────────────────────────────────────────────────┤
│                                                                  │
│  ┌──────────────┐                                                │
│  │ REQUIREMENT  │ ← New feature or bug ticket                    │
│  └──────┬───────┘                                                │
│         │                                                        │
│         ▼                                                        │
│  ┌──────────────┐                                                │
│  │ CREATE BRANCH│ git checkout -b feature/JIRA-123               │
│  └──────┬───────┘                                                │
│         │                                                        │
│         ▼                                                        │
│  ┌──────────────┐                                                │
│  │ WRITE TESTS  │ Create test files, page objects                │
│  └──────┬───────┘                                                │
│         │                                                        │
│         ▼                                                        │
│  ┌──────────────┐                                                │
│  │ TEST LOCALLY │ npx playwright test --headed                   │
│  └──────┬───────┘                                                │
│         │                                                        │
│      ┌───┴───┐                                                   │
│      │ Pass? │                                                   │
│      └───┬───┘                                                   │
│    No   │   Yes                                                  │
│    ↓    └────┐                                                   │
│  Fix bug     │                                                   │
│    ↑         │                                                   │
│    └─────────┤                                                   │
│              ▼                                                   │
│  ┌──────────────┐                                                │
│  │ COMMIT & PUSH│ git add . → commit → push                      │
│  └──────┬───────┘                                                │
│         │                                                        │
│         ▼                                                        │
│  ┌──────────────┐                                                │
│  │ CREATE PR    │ Pull Request to develop                        │
│  └──────┬───────┘                                                │
│         │                                                        │
│         ├──────────────┐                                         │
│         ▼              ▼                                         │
│  ┌──────────────┐ ┌──────────────┐                               │
│  │ CI RUNS      │ │ CODE REVIEW  │                               │
│  │ (Automatic)  │ │ (Reviewer)   │                               │
│  └──────┬───────┘ └──────┬───────┘                               │
│         │                │                                       │
│         └───────┬────────┘                                       │
│                 │                                                │
│         All checks pass?                                         │
│                 │                                                │
│            ┌────┴────┐                                           │
│           Yes       No                                           │
│            │         │                                           │
│            │      Fix issues                                     │
│            │         ↑                                           │
│            ▼         │                                           │
│  ┌──────────────┐────┘                                           │
│  │ MERGE TO     │                                                │
│  │ DEVELOP      │                                                │
│  └──────┬───────┘                                                │
│         │                                                        │
│         ▼                                                        │
│  ┌──────────────┐                                                │
│  │ CI RUNS      │ Full regression on develop                     │
│  │ REGRESSION   │                                                │
│  └──────┬───────┘                                                │
│         │                                                        │
│         ▼                                                        │
│  ┌──────────────┐                                                │
│  │ REPORTS      │ HTML reports available                         │
│  │ GENERATED    │                                                │
│  └──────┬───────┘                                                │
│         │                                                        │
│         ▼                                                        │
│  ┌──────────────┐                                                │
│  │ DELETE       │ Clean up feature branch                        │
│  │ BRANCH       │                                                │
│  └──────────────┘                                                │
│                                                                  │
│  ✅ WORKFLOW COMPLETE                                              │
│                                                                   │
└───────────────────────────────────────────────────────────────────┘
```

13.2 Quick Reference Commands

```
# ═══════════════════════════════════════════════════════════════
# DAILY WORKFLOW COMMANDS
# ═══════════════════════════════════════════════════════════════

# Start of day
git checkout develop
git pull origin develop

# Create feature branch
git checkout -b feature/JIRA-123-description

# Check status
git status

# Stage changes
git add .

# Commit with message
git commit -m "feat: add login test cases"

# Push branch
git push origin feature/JIRA-123-description

# ═══════════════════════════════════════════════════════════════
# TEST COMMANDS
# ═══════════════════════════════════════════════════════════════

# Run all tests
npx playwright test

# Run headed (see browser)
npx playwright test --headed

# Run specific file
npx playwright test tests/login.spec.ts

# Run by tag
npx playwright test --grep @smoke

# Debug mode
npx playwright test --debug
```

```
# Show report
npx playwright show-report

# ═══════════════════════════════════════════════════════════════
# AFTER MERGE
# ═══════════════════════════════════════════════════════════════

# Update local develop
git checkout develop
git pull origin develop

# Delete local branch
git branch -d feature/JIRA-123-description
```

13.3 Summary Checklist

Before Starting Work:

- [ ] Pull latest develop
- [ ] Create feature branch with proper naming
- [ ] Understand the requirement

While Working:

- [ ] Follow coding standards
- [ ] Write descriptive test names
- [ ] Use Page Object pattern
- [ ] No hardcoded values
- [ ] Add appropriate test tags

Before Pushing:

- [ ] Run tests locally
- [ ] All tests pass
- [ ] Code is clean (no console.log)
- [ ] Commit message follows standards

Pull Request:

- [ ] PR has descriptive title
- [ ] PR description explains changes

- [ ] Linked to ticket/requirement
- [ ] Assigned reviewers

After Merge:

- [ ] CI pipeline passes on develop
- [ ] Delete feature branch
- [ ] Update ticket status
- [ ] Notify team if needed

Continue to Bonus Section...