Testing & QA Guide

This guide covers all testing standards, tooling, accessibility verification, and responsive design checks required before any pull request is merged into Unicis Platform.

This page is part of the Development section.
Last updated: 2026-06-12

Testing Stack

Tool	Purpose	Command
Jest	Unit tests for utilities, hooks, and API handlers	npx jest
Playwright	E2E browser automation	npx playwright test
TypeScript	Type-safety gate	npx tsc –noEmit –skipLibCheck
ESLint	Code style and import checks	npx eslint .

Unit Tests — Jest

What to Test

- Pure utility functions (calculation helpers, formatters, mapping functions)
- API route handlers — test the handler function directly, stub Prisma and session
- React hooks with complex state logic
- Status/key regression guards (e.g. task status keys must match DB-stored values)

Coverage as of 2026-06-12

62 tests across:

- Status-key regression guard (task statuses inprogress/inreview match DB, not in-progress/in-review)
- CSC helper invariants
- Dashboard task-counting logic
- Full CRUD coverage for /api/teams/[slug]/tasks and /api/teams/[slug]/tasks/[taskNumber]
- Both CSC endpoints (single control update + bulk status change)

Writing a New Test

```
// __tests__/api/tasks.test.ts
import { createMocks } from 'node-mocks-http';
import handler from '@/pages/api/teams/[slug]/tasks';

test('GET returns 200 with task list', async () => {
  const { req, res } = createMocks({ method: 'GET' });
  await handler(req, res);
  expect(res._getStatusCode()).toBe(200);
```

```
});
```

Place test files in `tests/` mirroring the source path, or co-located as `*.test.ts`.

E2E Tests — Playwright

Suites as of 2026-06-12

Suite	File	What it covers
Dashboard	e2e/dashboard.spec.ts	Task matrix counts, tab switching (Data Protection / Cybersecurity / Risk Management)
Tasks	e2e/tasks.spec.ts	Task list view, Kanban board drag-and-drop, filter interactions
CSC	e2e/csc.spec.ts	Bulk status change across multiple controls, task assignment and refresh

Prerequisites

1. Local dev server running: npx next dev
2. Test database seeded with demo team + tasks + CSC controls
3. `.env.test` with `NEXTAUTH_URL=http://localhost:4002`

Running E2E Tests

```
npx playwright test              # all suites
npx playwright test e2e/csc.spec.ts   # single suite
npx playwright test --ui         # interactive UI mode
```

Accessibility Testing

accessibility_testing

Run the following checklist for every new or redesigned page before raising a PR.

Automated Checks

1. [ ] `npx tsc –noEmit` — no type errors that could mask missing aria props
2. [ ] Inspect DOM in browser DevTools → Accessibility panel — verify role tree is correct

Keyboard Navigation

1. [ ] Tab through every interactive element in logical order (no focus traps outside modals)
2. [ ] All buttons and links activate with Enter / Space
3. [ ] Modal dialogs: Tab stays inside; Escape closes; focus returns to trigger element
4. [ ] Tab bars: arrow keys move between tabs; Enter/Space selects

ARIA & Roles

1. [ ] `<html lang>` attribute present and matches page locale
2. [ ] Charts: wrapped in `role="img"` with meaningful `aria-label`
3. [ ] Tabs: `role="tablist"` on container, `role="tab"` on each tab, `role="tabpanel"` on each panel
4. [ ] Dialogs: `role="dialog"` + `aria-modal="true"` + `aria-labelledby` pointing to title
5. [ ] Icon-only buttons: `aria-label` describes the action
6. [ ] Decorative icons: `aria-hidden="true"`
7. [ ] Form error messages: input has `aria-invalid="true"` + `aria-describedby` pointing to error element
8. [ ] Error containers: `role="alert"` for immediate announcement

Colour Contrast

Use browser DevTools → CSS Overview or the WebAIM Contrast Checker:

1. [ ] Normal text ≥ 4.5:1 against background
2. [ ] Large text (≥ 18px or ≥ 14px bold) ≥ 3:1
3. [ ] UI component borders and icons ≥ 3:1
4. [ ] Never use `text-slate-400` for content text — minimum `text-slate-500`

Touch Targets

1. [ ] All buttons, links, and interactive elements are at least 44 × 44 CSS px
2. [ ] Verify using DevTools device emulation at 375 px width

Responsive Design Testing

Verify all three breakpoints using browser DevTools → device emulation:

Breakpoint	Viewport	Key checks
Mobile	375 × 812 px	Single column, stacked toolbar, horizontal table scroll, dialogs fit viewport
Tablet	768 × 1024 px	Two-column grids, tab bar visible, sidebar hidden
Desktop	1280 × 768 px	Full sidebar, multi-column grids, inline toolbars

Per-module Checklist

1. [ ] Dashboard — KPI row wraps to 2-col on mobile; tab bar scrolls; domain health cards stack
2. [ ] RPA / TIA / PIA — procedure table scrolls horizontally; dialogs use sticky-footer layout; Next button always visible
3. [ ] CSC — SectionRail hidden below lg; StatusesTable scrolls
4. [ ] RM — risk matrix chart wrapped in overflow-x-auto; dialog Next button always visible
5. [ ] IAP — course grid wraps; completion banner visible
6. [ ] All Tasks — toolbar stacks vertically on mobile; filter dropdowns share width evenly
7. [ ] Comments — single-column layout on all breakpoints

Pre-PR Checklist

Before requesting review on any pull request:

1. [ ] Jest tests pass: `npx jest`
2. [ ] Playwright E2E tests pass: `npx playwright test`
3. [ ] TypeScript clean: `npx tsc –noEmit –skipLibCheck`
4. [ ] Accessibility checklist above completed
5. [ ] Responsive design verified at 375 px, 768 px, 1280 px
6. [ ] All 7 locale files updated for any new translation keys
7. [ ] CHANGELOG entry added under the current release heading
8. [ ] `design.md` updated if a new component pattern was introduced

testing, QA, jest, playwright, accessibility, WCAG, responsive, i18n, checklist