Testing Standards

> Last Verified: 2026-02-17 | Owner: John

Testing Standard — GLOBAL (All Projects)

Authority: CEO directive (2026-02-13). Non-negotiable. **Scope:** Every project under ~/projects/, ~/ALAI/products/, and all client work.

ZAKON: No Ship Without Full Test Coverage

Nijedan projekat se NE DEPLOYA bez svih 5 nivoa testiranja. Svaki nivo mora proći MINIMUM 5 iteracija prije shipping-a.

Iteracija = full test run → fix failures → re-run. Prvih 5 iteracija otkrivaju 95% bugova.

---

5 Obaveznih Nivoa Testiranja

Nivo 1 — Unit Tests

Šta: Svaka funkcija, helper, utility, validator testiran izolovano. **Coverage:** Minimum 80% line coverage. **Alati:** Vitest (JS/TS), pytest (Python), Jest (ako već u projektu). **Pravilo:** Svaki novi fajl u `src/lib/`, `src/utils/`, `src/helpers/` MORA imati odgovarajući test fajl.

```
src/lib/auth.ts      → tests/unit/auth.test.ts
src/lib/utils.ts     → tests/unit/utils.test.ts
```

```
src/lib/middleware.ts → tests/unit/middleware.test.ts
```

Nivo 2 — Integration Tests

Šta: Moduli u kombinaciji — DB + API, Auth + Session, Payment + Rate. **Scope:** Svaki API endpoint testiran sa stvarnim DB-om (in-memory SQLite / test DB). **Pravilo:** Svaki API route MORA imati integration test koji:

- Šalje HTTP request (POST/GET/PUT/DELETE)
- Provjerava status code
- Provjerava response body strukturu
- Provjerava DB state nakon operacije
- Testira validation errors (bad input, missing fields, wrong types)
- Testira auth (unauthorized, forbidden)
- Testira edge cases (duplicate, not found, rate limit)

```
src/app/api/auth/register/route.ts → tests/integration/auth-register.test.ts
src/app/api/auth/login/route.ts    → tests/integration/auth-login.test.ts
src/app/api/transactions/*/route.ts → tests/integration/transactions.test.ts
```

Nivo 3 — E2E Tests (End-to-End)

Šta: Cijeli user flow od UI do DB i nazad. **Alati:** Playwright (primary), Cypress (alternativa). **Scope:** Svaki user-facing flow MORA imati e2e test:

- Registracija flow (form → API → success/error)
- Login flow (form → API → redirect → dashboard)
- Core feature flows (remittance, payment, etc.)
- Error flows (bad input → vidljiva error poruka)
- Navigation (svaka stranica loaduje, linkovi rade)

Pravilo: E2E testovi se pokreću protiv running dev server.

```
tests/e2e/registration.spec.ts
tests/e2e/login.spec.ts
tests/e2e/remittance.spec.ts
tests/e2e/navigation.spec.ts
```

Nivo 4 — Regression Tests

Šta: Testovi za SVAKI bug koji je pronađen i fixan. **Pravilo:** Kad se fix napravi, PRVO se napiše test koji reproducira bug, PA ONDA fix. **Format:** Test name mora sadržavati bug referencu.

```
it("BUG-001: rateLimit must be awaited to prevent bypass", async () => { ... });
it("BUG-002: validation errors must show specific details, not generic message", async () => {
... });
```

Direktorij: `tests/regression/`

Nivo 5 — Performance Tests (Ytelsestest)

Šta: Baseline performance za kritične operacije. **Alati:** k6, Artillery, ili custom Vitest benchmarks. **Scope:**

- API response time < 200ms (p95) za CRUD operacije
- Auth endpoints < 500ms (bcrypt je spor, to je OK)
- Concurrent users: minimum 50 simultanih sessiona
- DB query time: < 50ms za indexed queries
- Page load: < 3s FCP (First Contentful Paint)

```
tests/performance/api-latency.test.ts
tests/performance/concurrent-sessions.test.ts
tests/performance/page-load.test.ts
```

Test Execution Protocol

Prije svakog deploya/shippinga:

```
Iteracija 1: npm test                    → fix failures → commit
Iteracija 2: npm test                    → fix failures → commit
Iteracija 3: npm test                    → fix failures → commit
Iteracija 4: npm test                    → fix new edge cases → commit
Iteracija 5: npm test (FINAL)            → ALL GREEN → ready to ship
```

Svaka iteracija MORA biti logirana:

```
npm test 2>&1 | tee tests/logs/iteration-N.log
```

Test Coverage Report

Svaki projekat MORA generisati coverage report:

```
npx vitest run --coverage
```

Minimum pragovi:

- **Statements:** 80%
- **Branches:** 70%
- **Functions:** 80%
- **Lines:** 80%

Pipeline Gate — Testing Phase

Pipeline-controller.js Testing phase gate check:

1. ☐ `npm test` prolazi (exit code 0)
2. ☐ Postoje test fajlovi za svih 5 nivoa
3. ☐ Coverage >= 80%
4. ☐ Minimum 5 iteracija logirane
5. ☐ Zero KNOWN failures (sve što je nađeno je fixano)

Bez svih 5 — pipeline NE NAPREDUJE iz Testing faze.

Mandatory: Input Rejection Tests (Stupid User Category)

ZAKON: Svaki form input MORA imati testove koji provjeravaju da loš input bude ODBIJEN — ne samo da app "ne crashuje".

Problem koji je ovo pravilo izazvao: CEO je ručno ukucao "12345" u name polje i prošlo je. 765 linija chaos testova je provjeravalo "no crash" ali NIJEDAN test nije provjerio da forma ODBIJA loš input. 3-4 iteracije e2e testiranja — niko nije primijetio.

Dva tipa test assertions (OBA obavezna):

1. **Resilience assertion:** "App se ne ruši" → `expect(pageAlive).toBe(1)` — OVO NIJE DOVOLJNO
2. **Rejection assertion:** "App ODBIJA input" → `expect(errorVisible).toBe(true)` — OVO JE OBAVEZNO

Obavezni "stupid user" test inputi za SVAKO polje:

Polje tipa	Test input	Očekivano
Ime/prezime	`"12345"` (samo brojevi)	ODBIJENO — greška vidljiva
Ime/prezime	`"!@#$%"` (samo specijalni znakovi)	ODBIJENO — greška vidljiva
Ime/prezime	`"  "` (samo razmaci)	ODBIJENO — greška vidljiva
Email	`"notanemail"` (bez @)	ODBIJENO — greška vidljiva
Email	`"12345"` (brojevi)	ODBIJENO — greška vidljiva
Telefon	`"abcdef"` (slova umjesto brojeva)	ODBIJENO — greška vidljiva
Lozinka	`"12345678"` (samo brojevi, bez slova)	ODBIJENO — greška vidljiva
Datum	`"9999-99-99"` (nemoguć datum)	ODBIJENO — greška vidljiva
Iznos	`"-100"` (negativan)	ODBIJENO — greška vidljiva
Iznos	`"abc"` (tekst)	ODBIJENO — greška vidljiva
Bilo koje	`""` (prazno)	ODBIJENO — greška vidljiva

Pravilo za e2e chaos testove:

```
// □ NEDOVOLJNO — samo provjera da ne crashuje
const pageAlive = await page.locator("body").count();
expect(pageAlive).toBe(1);

// □ OBAVEZNO — provjera da je input ODBIJEN
const errorVisible = await page.locator('[class*="text-"][class*="EF4444"], [role="alert"]')
  .isVisible({ timeout: 3000 }).catch(() => false);
expect(errorVisible).toBe(true);
// I provjera da NIJE prošao na sljedeći korak
const nextStepVisible = await page.locator("text=next step indicator")
  .isVisible({ timeout: 1000 }).catch(() => false);
expect(nextStepVisible).toBe(false);
```

Princip:

> Ako imaš CHAOS_STRINGS definisane u testu — svaki string MORA biti testiran na SVAKOM polju. Inače, zašto postoji?

Project Test Structure (Template)

Svaki projekat MORA imati ovu strukturu:

```
tests/
├── unit/              ← Izolovani function testovi
├── integration/       ← API + DB testovi
├── e2e/               ← Full user flow testovi (Playwright)
├── regression/        ← Bug reproduction testovi
├── performance/       ← Latency, load, concurrent testovi
├── logs/              ← Iteration logs (iteration-1.log ... iteration-5.log)
└── coverage/          ← Coverage reports
```

Enforcement

Nivoi enforcement-a:

1. **Pipeline gate** — pipeline-controller.js blokira advance bez test evidence
2. **Pre-deploy hook** — blokira deploy ako `npm test` ne prolazi
3. **Code review** — reviewer MORA verificirati test coverage za svaki PR
4. **MC task** — testing task ne može biti DONE bez svih 5 nivoa

Consequences:

- App bez testova = app koja ne postoji
- "Radi na mom računaru" ≠ testirano
- Manual QA NIJE zamjena za automated tests (ali je DODATAK)

CEO Quote (2026-02-13):

>

> "Hocu da imamo testove za sve, svaki projekt. Ako nesto nije testirano — bicu pravo ljut. Unit test, integration test, e2e test, regression test, performance test. Ne ship app ako nije testirano sve i to 5 iteracija."

Ovo je zakon. Nema izuzetaka. Nema kompromisa.

Revision #4
Created 2026-02-17 22:14:44 UTC by John
Updated 2026-07-05 20:00:44 UTC by John