# E2E Test Plan

# E2E Test Plan

> **Project:** Bilko **Version:** 1.1 **Date:** 2026-05-21 **Author:** Ops Architect / ALAI Documentation Team **Status:** Active **Reviewers:** Tech Lead, Alem Bašić

## Document History

| Version | Date | Author | Changes |
|---|---|---|---|
| 0.1 | 2026-02-23 | Ops Architect | Initial draft |
| 1.0 | 2026-02-25 | ALAI Documentation Team | Finalized — approved for production use |
| 1.1 | 2026-05-21 | ALAI Documentation Team | Clarified critical E2E vs real-demo smoke vs full-demo rehearsal; documented non-destructive demo policy |

---

## 1. Overview

This plan covers Bilko's Playwright browser tests. E2E tests validate critical user journeys through a deployed application, but they are not intended to cover every behavior in the product. Financial correctness, API behavior, RBAC, and tenant isolation must be primarily proven by unit/integration/contract tests.

**Framework:** Playwright **Target:** small critical E2E suite + non-destructive real-demo smoke + scheduled full regression/demo rehearsal **Execution time:** < 8 minutes for critical gate; < 1 minute for real-demo smoke **Browsers:** Chromium primary for deploy gates; Firefox/WebKit on scheduled/risk-based runs **Base URL:** staging/resettable environment for critical/destructive E2E; `https://bilko-demo.alai.no` for non-destructive demo smoke

**Policy:** do not use the current full mixed Playwright suite as the deploy gate until it is partitioned. Specs that register users, create/delete data, torture rate limits, encode expected failures, or depend on one language must not run against public real demo as a blocking gate.

---

# 2. Test Environment

## Configuration

```
// apps/e2e/playwright.config.ts
import { defineConfig } from '@playwright/test'

export default defineConfig({
  testDir: './tests',
  timeout: 60000, // 60s per test
  retries: 1, // Retry flaky tests once
  workers: 4, // Parallel execution
  reporter: [['html'], ['github']],
  use: {
    baseURL: process.env.PLAYWRIGHT_BASE_URL || 'http://localhost:3000',
    screenshot: 'only-on-failure',
    video: 'retain-on-failure',
    trace: 'retain-on-failure',
  },
  projects: [
    { name: 'chromium', use: { browserName: 'chromium' } },
    { name: 'firefox', use: { browserName: 'firefox' } },
    { name: 'webkit', use: { browserName: 'webkit' } },
  ],
})
```

## Test Data

Use different data policies per environment:

- **Real public demo:** `demo@bilko.rs` / `Demo2026!`; read-only/non-destructive smoke only. No registration, no deletes, no rate-limit tests, no invoice/expense creation unless the tenant is explicitly resettable.

- **Critical staging E2E:** seeded resettable organization, user, customer, vendor, invoices, expenses, and reports. Tests may create/update data because the environment is disposable.
- **Full-demo rehearsal:** dedicated resettable demo tenant/environment with a scripted business story and automatic reset before each rehearsal.

Seed/reset commands must be implemented for staging/full-demo before destructive E2E is enabled as a gate.

---

# 3. E2E Test Suites

## Suite 0: Real Demo Smoke (non-destructive — P0)

**File:** `apps/e2e/tests/real-demo-smoke.spec.ts`

| ID | Test | Priority | Status |
|---|---|---|---|
| E2E-SMOKE-REAL | API health + login + protected page smoke | P0 | Implemented |

**Command:**

```
cd apps/e2e
npm run test:real-demo-smoke
```

Default targets:

- Frontend: `https://bilko-demo.alai.no`
- API: `https://bilko-demo-api.alai.no`

Coverage:

- API `/api/v1/health`
- API login with demo credentials
- refresh cookie/session validation
- `/dashboard`, `/invoices`, `/settings`
- unexpected browser console errors
- optional video evidence via `E2E_VIDEO_DIR`

Non-goals:

- registration
- invoice/expense creation
- deletes/status mutations
- PDF correctness
- multi-tenant isolation
- rate-limit torture
- historical expected-fail regressions

---

# Suite 1: Invoice Flow (4 tests — P0)

**File:** `apps/e2e/tests/invoice-flow.spec.ts`

| ID | Test | Priority | Status |
|---|---|---|---|
| E2E-INV-01 | Create invoice via 6-step wizard | P0 | Not implemented |
| E2E-INV-02 | Preview invoice before sending | P0 | Not implemented |
| E2E-INV-03 | Send invoice to customer via email | P0 | Not implemented |
| E2E-INV-04 | Mark invoice as paid | P0 | Not implemented |

### E2E-INV-01: Create invoice via 6-step wizard

```
test('create invoice via 6-step wizard', async ({ page }) => {
  await page.goto('/invoices/new')

  // Step 1: Customer selection
  await page.selectOption('[data-testid="customer-select"]', { label: 'Acme Corp' })
  await page.click('[data-testid="next-btn"]')

  // Step 2: Invoice details
  await page.fill('[data-testid="invoice-date"]', '2026-02-23')
  await page.fill('[data-testid="due-date"]', '2026-03-23')
  await page.selectOption('[data-testid="currency-select"]', 'RSD')
  await page.click('[data-testid="next-btn"]')

  // Step 3: Line items
  await page.fill('[data-testid="item-0-description"]', 'Web Development')
  await page.fill('[data-testid="item-0-quantity"]', '10')
  await page.fill('[data-testid="item-0-unit-price"]', '5000')
```

```
  await page.selectOption('[data-testid="item-0-tax-rate"]', '20')
  await page.click('[data-testid="next-btn"]')

  // Step 4: Review — verify totals
  await expect(page.locator('[data-testid="subtotal"]')).toContainText('50,000.00 RSD')
  await expect(page.locator('[data-testid="tax-amount"]')).toContainText('10,000.00 RSD')
  await expect(page.locator('[data-testid="total-amount"]')).toContainText('60,000.00 RSD')

  // Create
  await page.click('[data-testid="create-invoice-btn"]')

  // Verify redirect and invoice number
  await expect(page).toHaveURL(/\/invoices\/[a-f0-9-]+$/)
  await expect(page.locator('h1')).toContainText('INV-')
  await expect(page.locator('[data-testid="invoice-status"]')).toContainText('Draft')
})
```

**E2E-INV-04: Mark invoice as paid**

```
test('mark invoice as paid', async ({ page }) => {
  // Create invoice first (or use pre-seeded sent invoice)
  await page.goto('/invoices')
  await page.click('[data-testid="invoice-row"]:first-child')

  await page.click('[data-testid="mark-paid-btn"]')
  await page.fill('[data-testid="payment-date"]', '2026-02-23')
  await page.click('[data-testid="confirm-payment-btn"]')

  await expect(page.locator('[data-testid="invoice-status"]')).toContainText('Paid')
  await expect(page.locator('[data-testid="paid-at"]')).toBeVisible()
})
```

---

# Suite 2: Expense Flow (3 tests — P1)

**File:** `apps/e2e/tests/expense-flow.spec.ts`

| ID | Test | Priority | Status |
|---|---|---|---|
| E2E-EXP-01 | Add expense with receipt photo upload | P1 | Not implemented |

| ID | Test | Priority | Status |
|---|---|---|---|
| E2E-EXP-02 | Approve expense as admin | P1 | Not implemented |
| E2E-EXP-03 | Mark expense as paid | P1 | Not implemented |

### E2E-EXP-01: Add expense with receipt upload

```
test('add expense with receipt photo upload', async ({ page }) => {
  await page.goto('/expenses/new')

  await page.fill('[data-testid="expense-amount"]', '1500')
  await page.selectOption('[data-testid="expense-currency"]', 'RSD')
  await page.selectOption('[data-testid="expense-category"]', 'office_supplies')
  await page.fill('[data-testid="expense-description"]', 'Printer paper')
  await page.fill('[data-testid="expense-date"]', '2026-02-23')

  // Upload receipt image
  const fileInput = page.locator('[data-testid="receipt-upload"]')
  await fileInput.setInputFiles('fixtures/test-receipt.jpg')
  await expect(page.locator('[data-testid="receipt-preview"]')).toBeVisible()

  await page.click('[data-testid="submit-expense-btn"]')

  await expect(page).toHaveURL(/\/expenses\/[a-f0-9-]+$/)
  await expect(page.locator('[data-testid="expense-status"]')).toContainText('Pending')
})
```

---

# Suite 3: Report Flow (2 tests — P1)

**File:** `apps/e2e/tests/report-flow.spec.ts`

| ID | Test | Priority | Status |
|---|---|---|---|
| E2E-REP-01 | Generate P&L report for date range | P1 | Not implemented |
| E2E-REP-02 | Export P&L report to PDF | P1 | Not implemented |

### E2E-REP-01: Generate P&L report

```
test('generate P&L report for date range', async ({ page }) => {
  await page.goto('/reports/profit-loss')
```

```
  await page.fill('[data-testid="start-date"]', '2026-01-01')
  await page.fill('[data-testid="end-date"]', '2026-01-31')
  await page.click('[data-testid="generate-btn"]')

  await expect(page.locator('[data-testid="report-title"]')).toContainText('Profit & Loss')
  await expect(page.locator('[data-testid="total-revenue"]')).toBeVisible()
  await expect(page.locator('[data-testid="total-expenses"]')).toBeVisible()
  await expect(page.locator('[data-testid="net-profit"]')).toBeVisible()
})
```

---

# Suite 4: Auth Flow (1 test — P1)

**File:** `apps/e2e/tests/auth-flow.spec.ts`

| ID | Test | Priority | Status |
|---|---|---|---|
| E2E-AUTH-01 | Register → Login → 2FA → Logout | P1 | Not implemented |

### E2E-AUTH-01: Full auth flow

```
test('register, login with 2FA, and logout', async ({ page }) => {
  // Register new user
  await page.goto('/register')
  await page.fill('[data-testid="company-name"]', 'Test Firma d.o.o.')
  await page.fill('[data-testid="email"]', `test_${Date.now()}@bilko.io`)
  await page.fill('[data-testid="password"]', 'SecurePass123!')
  await page.click('[data-testid="register-btn"]')

  // Should redirect to dashboard after registration
  await expect(page).toHaveURL('/dashboard')

  // Logout
  await page.click('[data-testid="user-menu"]')
  await page.click('[data-testid="logout-btn"]')
  await expect(page).toHaveURL('/login')
})
```

---

## Suite 5: Settings Flow (2 tests — P2)

**File:** `apps/e2e/tests/settings-flow.spec.ts`

| ID | Test | Priority | Status |
|---|---|---|---|
| E2E-SET-01 | Update organization settings | P2 | Not implemented |
| E2E-SET-02 | Invite user to organization | P2 | Not implemented |

---

# 4. Test Execution

## Running E2E Tests

```
cd apps/e2e

# Non-destructive public demo smoke — deploy/demo health gate
npm run test:real-demo-smoke

# All Playwright specs — developer/regression use only until partitioned
npm test

# Specific browser
npm test -- --project=chromium
npm test -- --project=firefox
npm test -- --project=webkit

# Headed/debug modes
npm run test:headed
npm test -- --debug

# Single test file
npm test -- tests/invoices.spec.ts --project=chromium

# Generate HTML report
npm test -- --reporter=html
```

When running against real demo with video evidence:

```
E2E_REAL_DEMO_SMOKE=1 \
E2E_BASE_URL=https://bilko-demo.alai.no \
E2E_API_URL=https://bilko-demo-api.alai.no \
E2E_VIDEO_DIR=/tmp/alai/bilko-real-demo-smoke-video-$(date +%Y%m%d-%H%M%S)/videos \
npm test -- tests/real-demo-smoke.spec.ts --project=chromium --no-deps
```

## On CI/CD

Recommended gates:

```
# Staging/resettable environment: critical E2E only
# Add this script after Playwright specs are tagged/partitioned.
E2E_BASE_URL=${STAGING_WEB_URL} E2E_API_URL=${STAGING_API_URL} npm run test:e2e:critical

# Post-deploy/public demo: non-destructive smoke only — implemented now.
E2E_BASE_URL=https://bilko-demo.alai.no E2E_API_URL=https://bilko-demo-api.alai.no npm run
test:real-demo-smoke
```

Artifacts: result JSON, screenshots, traces/videos where enabled. Evidence should be stored outside the repo, e.g. `/tmp/alai/<run-id>/` or CI artifacts.

---

# 5. Flaky Test Policy

1. Retry failed critical E2E test once (`retries: 1` in Playwright config where enabled)
2. If still fails: mark as `flaky` in Mission Control/GitHub Issues, do NOT ignore
3. Fix flaky critical tests within the same sprint they appear
4. Quarantine only non-blocking nightly/regression tests; never silently skip deploy-gate tests
5. Common causes: race conditions (use `await expect()` not `await sleep()`), timing, localization assumptions, shared auth state, and test data isolation

---

# 6. Accessibility Checks (PLANNED)

Add Axe accessibility checks to critical flows:

```
import { checkA11y } from 'axe-playwright'
```

```
test('invoice form is accessible', async ({ page }) => {
  await page.goto('/invoices/new')
  await checkA11y(page, '#invoice-form', {
    detailedReport: true,
    detailedReportOptions: { html: true },
  })
})
```

# Related Documents

- Test Strategy
- Test Plan
- TESTING-GUIDE.md
- CI/CD Pipeline
- Demo Testing Plan

# Approval

| Role | Name | Date | Signature |
|---|---|---|---|
| Author | Ops Architect | 2026-02-23 | |
| Reviewer | Tech Lead | | |
| Approver | Alem Bašić | | |

Revision #17
Created 2026-02-24 22:50:55 UTC by John
Updated 2026-06-07 19:43:40 UTC by John