Test Inventory

Bilko — Test Inventory

Status: NO TESTS EXIST YET — This document tracks planned tests and implementation status.

This inventory catalogs all tests planned for Bilko, organized by category and priority.

Test Coverage Summary

Module	Unit Planned	Integration Planned	E2E Planned	Unit Target	Int Target
Financial logic (`@bilko/core`)	35	—	—	**95%**	N/A
Auth (JWT, RBAC, 2FA)	8	10	1	85%	**90%**
Invoicing (CRUD, lifecycle, tax)	12	10	2	85%	**90%**
Expenses (CRUD, approval, payment)	6	8	1	80%	**85%**
Banking (import, reconciliation)	4	8	1	75%	**75%**
Reports (P&L, VAT, balance sheet)	6	7	1	80%	**80%**
Multi-tenant isolation	—	8	—	N/A	**100%**
Frontend UI components	—	—	12	N/A	N/A — **70%**
TOTAL	**71**	**51**	**18**	—	—

Priority Legend

- **P0** — Critical (MVP blocker, financial logic)
- **P1** — High (core features, security)
- **P2** — Medium (nice-to-have, edge cases)
- **P3** — Low (future enhancements)

Unit Tests (45 total)

Financial Calculations (12 tests)

Test File	Test Name	What It Tests	Priority	Status
`vat.test.ts`	`calculateVAT - Serbia 20%`	VAT calculation for Serbia	P0	⬜ Not implemented
`vat.test.ts`	`calculateVAT - BiH 17%`	VAT calculation for BiH	P0	⬜ Not implemented
`vat.test.ts`	`calculateVAT - Croatia 25%`	VAT calculation for Croatia	P0	⬜ Not implemented
`vat.test.ts`	`calculateVAT - zero rate`	VAT at 0% (exports)	P0	⬜ Not implemented
`vat.test.ts`	`calculateVAT - decimal amounts`	VAT on €123.45	P0	⬜ Not implemented
`vat.test.ts`	`calculateVAT - rounding`	Rounds to 2 decimal places	P0	⬜ Not implemented
`invoice-calc.test.ts`	`calculateInvoiceTotal - subtotal`	Sum of line items	P0	⬜ Not implemented
`invoice-calc.test.ts`	`calculateInvoiceTotal - tax`	Sum of tax amounts	P0	⬜ Not implemented
`invoice-calc.test.ts`	`calculateInvoiceTotal - discount`	Subtract discount from subtotal	P0	⬜ Not implemented
`invoice-calc.test.ts`	`calculateInvoiceTotal - total`	Subtotal + tax - discount	P0	⬜ Not implemented
`invoice-calc.test.ts`	`calculateInvoiceTotal - multi-item`	Multiple line items with different tax rates	P0	⬜ Not implemented
`invoice-calc.test.ts`	`calculateInvoiceTotal - precision`	NUMERIC(19,4) precision maintained	P0	⬜ Not implemented

---

Double-Entry Validation (6 tests)

Test File	Test Name	What It Tests	Priority	Status

`double-entry.test.ts`	`validateTransaction - debit equals credit`	Debit amount = Credit amount	P0	⬜ Not implemented
`double-entry.test.ts`	`validateTransaction - rejects unbalanced`	Throws error if debit ≠ credit	P0	⬜ Not implemented
`double-entry.test.ts`	`validateTransaction - requires both accounts`	Throws if missing debit or credit account	P0	⬜ Not implemented
`double-entry.test.ts`	`validateTransaction - multi-currency`	Validates amounts in base currency	P0	⬜ Not implemented
`double-entry.test.ts`	`validateTransaction - precision`	NUMERIC precision preserved	P0	⬜ Not implemented
`double-entry.test.ts`	`validateTransaction - zero amount`	Rejects zero-amount transactions	P1	⬜ Not implemented

Currency Conversion (8 tests)

Test File	Test Name	What It Tests	Priority	Status
`currency.test.ts`	`convertCurrency - EUR to RSD`	Convert at locked exchange rate	P0	⬜ Not implemented
`currency.test.ts`	`convertCurrency - RSD to EUR`	Reverse conversion	P0	⬜ Not implemented
`currency.test.ts`	`convertCurrency - same currency`	Rate = 1.0 when currency matches	P0	⬜ Not implemented
`currency.test.ts`	`convertCurrency - precision`	NUMERIC(19,4) preserved	P0	⬜ Not implemented
`currency.test.ts`	`convertCurrency - large amounts`	€999,999,999.9999	P0	⬜ Not implemented
`currency.test.ts`	`convertCurrency - rounding`	Rounds to 4 decimal places	P1	⬜ Not implemented
`currency.test.ts`	`lockExchangeRate - historical rate`	Uses rate from transaction date, not today	P0	⬜ Not implemented
`currency.test.ts`	`lockExchangeRate - missing rate`	Throws if no rate available for date	P1	⬜ Not implemented

Date Utilities (6 tests)

Test File	Test Name	What It Tests	Priority	Status
`date.test.ts`	`calculateDueDate - 30 days`	Invoice date + 30 days	P1	⬜ Not implemented

Test File	Test Name	What It Tests	Priority	Status
`date.test.ts`	`calculateDueDate - custom terms`	Invoice date + custom days	P1	⬜ Not implemented
`date.test.ts`	`isOverdue - past due date`	Returns true if today > due date	P1	⬜ Not implemented
`date.test.ts`	`isOverdue - not overdue`	Returns false if today <= due date	P1	⬜ Not implemented
`date.test.ts`	`getFiscalYear - starts Jan 1`	Fiscal year 2026 = Jan 1 - Dec 31	P2	⬜ Not implemented
`date.test.ts`	`getFiscalYear - custom start`	Fiscal year starts on custom date	P2	⬜ Not implemented

Number Formatting (5 tests)

Test File	Test Name	What It Tests	Priority	Status
`format.test.ts`	`formatCurrency - RSD`	"50,000.00 RSD" format	P1	⬜ Not implemented
`format.test.ts`	`formatCurrency - EUR`	"€50,000.00" format	P1	⬜ Not implemented
`format.test.ts`	`formatCurrency - BAM`	"50,000.00 BAM" format	P1	⬜ Not implemented
`format.test.ts`	`formatCurrency - decimal places`	Respects currency decimal places (0-4)	P1	⬜ Not implemented
`format.test.ts`	`formatCurrency - null`	Returns "-" for null/undefined	P2	⬜ Not implemented

Authentication (8 tests)

Test File	Test Name	What It Tests	Priority	Status
`auth.test.ts`	`hashPassword - bcrypt 12 rounds`	Password hashed with bcrypt	P0	⬜ Not implemented
`auth.test.ts`	`hashPassword - unique salt`	Each hash is different	P0	⬜ Not implemented
`auth.test.ts`	`verifyPassword - correct password`	Returns true for correct password	P0	⬜ Not implemented
`auth.test.ts`	`verifyPassword - incorrect password`	Returns false for wrong password	P0	⬜ Not implemented
`auth.test.ts`	`generateJWT - valid payload`	JWT contains user ID, org ID, role	P0	⬜ Not implemented

Test File	Test Name	What It Tests	Priority	Status
`auth.test.ts`	`generateJWT - expiry 15 min`	Access token expires in 15 min	P0	⬜ Not implemented
`auth.test.ts`	`generateRefreshToken - expiry 7 days`	Refresh token expires in 7 days	P0	⬜ Not implemented
`auth.test.ts`	`verifyJWT - expired token`	Throws error if token expired	P1	⬜ Not implemented

Integration Tests (35 total)

Auth API (10 tests)

Test File	Test Name	What It Tests	Priority	Status
`auth-api.test.ts`	`POST /auth/register - success`	Creates user, returns 201	P0	⬜ Not implemented
`auth-api.test.ts`	`POST /auth/register - duplicate email`	Returns 400 if email exists	P0	⬜ Not implemented
`auth-api.test.ts`	`POST /auth/register - weak password`	Returns 400 if password < 8 chars	P0	⬜ Not implemented
`auth-api.test.ts`	`POST /auth/login - success`	Returns access + refresh tokens	P0	⬜ Not implemented
`auth-api.test.ts`	`POST /auth/login - wrong password`	Returns 401	P0	⬜ Not implemented
`auth-api.test.ts`	`POST /auth/login - non-existent user`	Returns 401	P0	⬜ Not implemented
`auth-api.test.ts`	`POST /auth/refresh - success`	Returns new access token	P0	⬜ Not implemented
`auth-api.test.ts`	`POST /auth/refresh - expired token`	Returns 401	P1	⬜ Not implemented
`auth-api.test.ts`	`POST /auth/logout - success`	Deletes refresh token from DB	P1	⬜ Not implemented
`auth-api.test.ts`	`POST /auth/logout - already logged out`	Returns 204 (idempotent)	P2	⬜ Not implemented

Invoices API (10 tests)

Test File	Test Name	What It Tests	Priority	Status

`invoices-api.test.ts`	`POST /invoices - success`	Creates invoice, returns 201	P0	⬜ Not implemented
`invoices-api.test.ts`	`POST /invoices - validates required fields`	Returns 400 if missing customer	P0	⬜ Not implemented
`invoices-api.test.ts`	`POST /invoices - validates currency`	Returns 400 if invalid currency code	P0	⬜ Not implemented
`invoices-api.test.ts`	`POST /invoices - org scoping`	Returns 403 if customer in different org	P0	⬜ Not implemented
`invoices-api.test.ts`	`GET /invoices - list`	Returns paginated invoices	P0	⬜ Not implemented
`invoices-api.test.ts`	`GET /invoices - filter by status`	Returns only "paid" invoices	P1	⬜ Not implemented
`invoices-api.test.ts`	`GET /invoices/:id - success`	Returns invoice by ID	P0	⬜ Not implemented
`invoices-api.test.ts`	`GET /invoices/:id - not found`	Returns 404 if ID doesn't exist	P1	⬜ Not implemented
`invoices-api.test.ts`	`PATCH /invoices/:id - update status`	Changes status to "sent"	P0	⬜ Not implemented
`invoices-api.test.ts`	`DELETE /invoices/:id - soft delete`	Marks as deleted (not hard delete)	P1	⬜ Not implemented

Expenses API (8 tests)

Test File	Test Name	What It Tests	Priority	Status
`expenses-api.test.ts`	`POST /expenses - success`	Creates expense, returns 201	P0	⬜ Not implemented
`expenses-api.test.ts`	`POST /expenses - validates required fields`	Returns 400 if missing amount	P0	⬜ Not implemented
`expenses-api.test.ts`	`POST /expenses - org scoping`	Returns 403 if vendor in different org	P0	⬜ Not implemented
`expenses-api.test.ts`	`GET /expenses - list`	Returns paginated expenses	P0	⬜ Not implemented
`expenses-api.test.ts`	`PATCH /expenses/:id/approve - success`	Changes status to "approved"	P1	⬜ Not implemented
`expenses-api.test.ts`	`PATCH /expenses/:id/approve - requires admin`	Returns 403 if user is viewer	P1	⬜ Not implemented
`expenses-api.test.ts`	`PATCH /expenses/:id/reject - success`	Changes status to "rejected"	P1	⬜ Not implemented

Test File	Test Name	What It Tests	Priority	Status
`expenses-api.test.ts`	`DELETE /expenses/:id - soft delete`	Marks as deleted	P1	⬜ Not implemented

Reports API (7 tests)

Test File	Test Name	What It Tests	Priority	Status
`reports-api.test.ts`	`GET /reports/profit-loss - success`	Returns P&L with revenue, expenses, net	P1	⬜ Not implemented
`reports-api.test.ts`	`GET /reports/profit-loss - date range`	Filters by start/end date	P1	⬜ Not implemented
`reports-api.test.ts`	`GET /reports/balance-sheet - success`	Returns assets, liabilities, equity	P1	⬜ Not implemented
`reports-api.test.ts`	`GET /reports/cash-flow - success`	Returns operating, investing, financing	P1	⬜ Not implemented
`reports-api.test.ts`	`GET /reports/vat - success`	Returns sales VAT, purchase VAT, net VAT	P1	⬜ Not implemented
`reports-api.test.ts`	`GET /reports/vat - Serbia 20%`	Calculates Serbian VAT correctly	P1	⬜ Not implemented
`reports-api.test.ts`	`GET /reports/vat - export PDF`	Returns PDF file	P2	⬜ Not implemented

Banking & Reconciliation Tests (Integration — 8 tests)

Test File	Test Name	What It Tests	Priority	Status
`banking-api.test.ts`	`POST /bank-accounts/:id/import - success`	Parses CSV, returns imported count	P0	⬜ Not implemented
`banking-api.test.ts`	`POST /bank-accounts/:id/import - duplicate detection`	Skips already-imported transactions	P0	⬜ Not implemented
`banking-api.test.ts`	`POST /bank-accounts/:id/import - invalid CSV format`	Returns 400 with validation errors	P1	⬜ Not implemented

Test File	Test Name	What It Tests	Priority	Status
`banking-api.test.ts`	`GET /bank-accounts/:id/transactions - lists unreconciled`	Returns unreconciled bank txs	P1	⬜ Not implemented
`banking-api.test.ts`	`POST /bank-accounts/:id/reconcile - matches bank tx to GL`	Sets reconciled=true on both	P0	⬜ Not implemented
`banking-api.test.ts`	`POST /bank-accounts/:id/reconcile - amount mismatch`	Returns 400 if amounts differ	P0	⬜ Not implemented
`banking-api.test.ts`	`POST /bank-accounts/:id/reconcile - cross-org bank tx`	Returns 404, not 403 (prevent enumeration)	P0	⬜ Not implemented
`banking-api.test.ts`	`GET /bank-accounts - org scoping`	Returns only current org bank accounts	P0	⬜ Not implemented

---

Financial Calculation Edge Cases (Unit — critical)

These edge cases specifically target financial precision issues that affect real money and compliance.

NUMERIC(19,4) Precision Tests

Test Name	Input	Expected Output	Risk if Fails
No float drift: 0.1 + 0.2 = 0.3	`Decimal('0.1').plus('0.2')`	`'0.30'`	Invoice totals miscalculated
Large amount precision: 999,999,999.9999 × 1.20	`Decimal('999999999.9999').mul('1.20')`	`'1199999999.9999'`	Multi-million invoices wrong
4 decimal place storage: 1/3	`Decimal('1').div('3').toDP(4)`	`'0.3333'`	Unit price rounding error
Penny rounding: 10.005 × 20%	`Decimal('10.005').mul('0.20').toDP(4)`	`'2.0010'`	VAT line items off by 1 cent
Accumulated rounding: sum of 100 items @ 0.01	Sum 100 × `Decimal('0.01')`	`'1.0000'`	Invoice total rounding drift
Multi-currency conversion precision	1 EUR × 117.1234 rate	`'117.1234'`	FX amount truncation

Multi-Currency Edge Cases

Test Name	Scenario	Expected Behavior
Exchange rate locked at invoice date	Rate changes after invoice creation	Invoice uses original rate, not current
Missing exchange rate on invoice date	No rate in ExchangeRate table for that date	Returns 422: rate unavailable, cannot create
Same-currency invoice	Invoice currency = org base currency	`exchangeRate = 1.000000`, `baseAmount = totalAmount`
Zero exchange rate rejected	Rate field = 0	Throws "exchange rate must be positive"
Forex gain/loss on payment	Invoice: 1000 EUR @ 117. Paid: 1000 EUR @ 120	Forex gain = (120-117) × 1000 = 3000 RSD
Penny rounding in multi-currency	100.005 EUR × 117.1234	Base = `11712.3457` (round half-up to 4dp)

VAT / Tax Edge Cases

Test Name	Scenario	Expected Behavior
Mixed tax rates on one invoice	Line 1: 20%, Line 2: 10%, Line 3: 0%	Each line has separate tax amount; totals sum correctly
Zero-rate export	Item exported outside RS/BA/HR	Tax rate 0%, taxAmount = 0, documented as zero-rated
Reverse VAT (gross to net)	Gross 1170 BAM at 17%	Net = 1000, VAT = 170 — not Net = 999.99
Croatian multiple VAT rates on one invoice	Lines at 25%, 13%, 5%	Each calculated independently; VAT report groups by rate
VAT report: draft invoices excluded	Draft invoice with tax	Does NOT appear in output VAT total
Penny rounding BiH: 7 items × 14.29 BAM × 17%	7 × 14.29 = 100.03, VAT = 17.005	VAT = 17.0051 (4dp), display rounds to 17.01

Penny Rounding (Banker's Rounding)

```
// Bilko uses ROUND_HALF_EVEN (banker's rounding) per accounting standards
describe('Banker\'s rounding (ROUND_HALF_EVEN)', () => {
  it('rounds 2.5 down to 2 (half-even)', () => {
    expect(new Decimal('2.5').toDecimalPlaces(0,
Decimal.ROUND_HALF_EVEN).toString()).toBe('2');
  });
```

```
  it('rounds 3.5 up to 4 (half-even)', () => {
    expect(new Decimal('3.5').toDecimalPlaces(0,
Decimal.ROUND_HALF_EVEN).toString()).toBe('4');
  });

  it('does NOT use ROUND_HALF_UP (which accumulates bias over many invoices)', () => {
    // ROUND_HALF_UP on 2.5 → 3 (biased upward)
    // ROUND_HALF_EVEN on 2.5 → 2 (unbiased)
    const items = ['0.5', '1.5', '2.5', '3.5', '4.5'];
    const sumHalfEven = items.reduce(
      (sum, v) => sum.plus(new Decimal(v).toDecimalPlaces(0, Decimal.ROUND_HALF_EVEN)),
      new Decimal(0)
    );
    // 0 + 2 + 2 + 4 + 4 = 12 (unbiased)
    expect(sumHalfEven.toString()).toBe('12');
  });
});
```

E2E Tests (18 total)

Invoice Flow (4 tests)

Test File	Test Name	What It Tests	Priority	Status
`invoice-flow.spec.ts`	`Create invoice via 6-step wizard`	Full invoice creation flow	P0	⬜ Not implemented
`invoice-flow.spec.ts`	`Preview invoice before sending`	Preview modal shows correct totals	P0	⬜ Not implemented
`invoice-flow.spec.ts`	`Send invoice to customer`	Email sent, status changed to "sent"	P0	⬜ Not implemented
`invoice-flow.spec.ts`	`Mark invoice as paid`	Status changed to "paid", paidAt timestamp	P0	⬜ Not implemented

---

Expense Flow (3 tests)

Test File	Test Name	What It Tests	Priority	Status

`expense-flow.spec.ts`	`Add expense with receipt upload`	Create expense, upload JPG	P1	⬜ Not implemented
`expense-flow.spec.ts`	`Approve expense`	Admin approves, status changed	P1	⬜ Not implemented
`expense-flow.spec.ts`	`Mark expense as paid`	Status changed to "paid"	P1	⬜ Not implemented

Report Flow (2 tests)

Test File	Test Name	What It Tests	Priority	Status
`report-flow.spec.ts`	`Generate P&L report`	Select date range, view report	P1	⬜ Not implemented
`report-flow.spec.ts`	`Export P&L to PDF`	Download PDF file	P1	⬜ Not implemented

Settings Flow (2 tests)

Test File	Test Name	What It Tests	Priority	Status
`settings-flow.spec.ts`	`Update organization settings`	Change org name, tax settings	P2	⬜ Not implemented
`settings-flow.spec.ts`	`Invite user to organization`	Send invite email, user accepts	P2	⬜ Not implemented

---

Auth Flow (1 test)

Test File	Test Name	What It Tests	Priority	Status
`auth-flow.spec.ts`	`Register → Login → 2FA → Logout`	Full auth flow with 2FA	P1	⬜ Not implemented

Banking Flow (3 tests)

Test File	Test Name	What It Tests	Priority	Status
`banking-flow.spec.ts`	`Import bank statement CSV`	Upload CSV, view imported transactions	P1	⬜ Not implemented
`banking-flow.spec.ts`	`Match bank transaction to invoice payment`	Reconcile bank entry with GL entry	P1	⬜ Not implemented

Test File	Test Name	What It Tests	Priority	Status
`banking-flow.spec.ts`	`Duplicate import shows correct warning`	Re-import same CSV shows 0 new imports	P2	Not implemented

Accessibility (3 tests)

Test File	Test Name	What It Tests	Priority	Status
`accessibility.spec.ts`	`Dashboard passes axe-core audit`	No WCAG 2.1 AA violations on dashboard	P1	Not implemented
`accessibility.spec.ts`	`Invoice create form passes axe-core audit`	No WCAG violations on form wizard	P1	Not implemented
`accessibility.spec.ts`	`Reports page passes axe-core audit`	No WCAG violations on reports view	P2	Not implemented

---

Test Implementation Roadmap

Phase 1 (MVP Critical) — 25 tests

- **Financial calculations** (12 unit tests)
- **Double-entry validation** (6 unit tests)
- **Auth API** (7 integration tests: register, login, refresh)

Target: Before backend MVP launch

Phase 2 (Core Features) — 35 tests

- **Currency conversion** (8 unit tests)
- **Invoices API** (10 integration tests)
- **Invoice E2E flow** (4 E2E tests)
- **Auth E2E flow** (1 E2E test)
- **Expense flow** (3 E2E tests)
- **Reports API** (7 integration tests)
- **Report E2E flow** (2 E2E tests)

Target: 1 month after MVP launch

Phase 3 (Polish) — 32 tests

- □ **Date utilities** (6 unit tests)
- □ **Number formatting** (5 unit tests)
- □ **Expenses API** (8 integration tests)
- □ **Settings flow** (2 E2E tests)
- □ **Remaining auth tests** (3 integration tests)
- □ **Edge cases** (8 tests across categories)

Target: 3 months after MVP launch

Test Execution Commands

Run All Tests

```
npm run test        # All tests (unit + integration + E2E)
```

Run by Category

```
npm run test:unit           # Unit tests only
npm run test:integration    # Integration tests only
npm run test:e2e            # E2E tests only
```

Run Specific Test File

```
npm run test:unit -- vat.test.ts
npm run test:integration -- invoices-api.test.ts
npm run test:e2e -- invoice-flow.spec.ts
```

Run with Coverage

```
npm run test:unit -- --coverage
```

Watch Mode

```
npm run test:unit -- --watch
```

Coverage Tracking

Current Coverage (as of 2026-02-20)

Category	Coverage	Target	Status
Financial Logic	0%	>95%	⏳ Not started
API Endpoints	0%	>80%	⏳ Not started
Utilities	0%	>90%	⏳ Not started
Overall	0%	>80%	⏳ Not started

Next Milestone: 50% coverage (25 critical tests)

Related Documents

- Testing Guide: TESTING-GUIDE.md
- CI/CD Pipeline: ../infrastructure/CI-CD.md
- Security Testing: ../security/SECURITY-ARCHITECTURE.md

Last Updated: 2026-02-20 **Status:** NO TESTS IMPLEMENTED YET **Total Tests Planned:** 92 (45 unit + 35 integration + 12 E2E) **Next Action:** Implement Phase 1 financial calculation tests (12 tests)

Revision #3
Created 2026-02-24 22:50:56 UTC by John
Updated 2026-05-31 20:04:03 UTC by John