

Behavioral Testing Guide

Finanzas SD – Architecture, Flows & SOPs

Arquitectura, Flujos y Procedimientos

December 17, 2025

1 Behavioral Testing Guide

This document describes the behavioral testing protocol for the Finanzas SD application, including API tests, UI tests, and security validations.

1.1 Overview

The behavioral testing protocol validates: 1. **API Reachability & Auth** - Cognito authentication works and API responds 2. **CORS Preflight** - CloudFront SPA origin is properly configured 3. **RBAC Enforcement** - Backend enforces role-based access control 4. **UI Behavior** - Frontend routing, navigation, and data state handling 5. **Security** - NO_GROUP users are denied access (prevents role leakage) 6. **Data Shape Contracts** - API responses match expected schemas

1.2 Key Principles

- **No Seeded Data Required** - Tests discover real data and handle empty states
- **Real Authentication** - Tests use actual Cognito users with different roles
- **Behavioral Validation** - Tests validate behavior, not implementation details
- **Security First** - NO_GROUP user validation prevents unauthorized access

1.3 Test Suites

1.3.1 1. Behavioral API Tests (Blocking on PR)

Location: tests/behavioral-api/

These tests run on every PR and validate:

Health/Smoke Tests (health.test.ts)

- API base URL is configured and reachable
- Cognito authentication works
- JSON responses are parseable
- CORS headers present on actual requests

CORS Preflight Tests (cors.test.ts)

- OPTIONS preflight requests return correct headers
- CloudFront origin is allowed
- Required methods (GET, POST, PATCH) are allowed
- Required headers (authorization, content-type) are allowed
- Tests all critical endpoints:
 - /projects

- /plan/forecast
- /catalog/rubros
- /line-items
- /allocation-rules
- /providers

RBAC Enforcement Tests (rbac.test.ts)

- **PMO** users can access PMO endpoints
- **SDM_FIN** users can access finance endpoints
- **SDMT** users can access SDMT endpoints
- **EXEC_RO** users have read-only access
- **NO_GROUP** users are denied all protected endpoints (CRITICAL)

Schema Validation Tests (schema.test.ts)

- /projects returns { data: array, total: number }
- Each project has projectId, code, name, client
- /plan/forecast returns { projectId, data: array }
- /catalog/rubros returns array of rubros
- Empty responses handled gracefully (no 500 errors)

1.3.2 2. Playwright UI Tests (Non-blocking, Manual/Nightly)

Location: tests/e2e/behavioral/

These tests validate UI behavior:

Routing RBAC Tests (routing-rbac.spec.ts)

- **NO_GROUP** users redirected or denied access
- **SDMT** users can access SDMT routes
- **SDM_FIN** users can access finance routes
- **EXEC_RO** users see all routes but actions are disabled
- **PMO** users blocked from SDMT-only routes

UI Data State Tests (ui-data-state.spec.ts)

- Empty states shown clearly when no data
- Charts don't render misleading empty visuals
- Populated data displayed correctly
- Loading indicators shown during fetch
- Error states handled gracefully

1.4 Running Tests Locally

1.4.1 Prerequisites

1. Environment Variables

Create `.env.local` with:

```
[ ] # API Configuration FINZ_API_BASE=https://pyorjw6lbe.execute-api.us-east-2.amazonaws.com
CF_DOMAIN=https://d7t9x3j66yd8k.cloudfront.net
# Cognito Configuration AWS_REGION=us-east-2 COGNITO_USER_POOL_ID=us-east-
2_FyHLtOhiY COGNITO_WEB_CLIENT=dshos5iou44tuach7ta3ici5m COGNITO_REGION=us-
east-2
# UI Configuration (for Playwright tests) FINZ_UI_BASE_URL=https://d7t9x3j66yd8k.cloudfront.net
```

2. Test Credentials

For behavioral tests, configure credentials for each role:

```
[ ] # PMO Role E2E_PMO_EMAIL=pmo-user@example.com E2E_PMO_PASSWORD=SecurePassword
# SDMT Role (also used for SDM_FIN tests) E2E_SDMT_EMAIL=sdmt-user@example.com
E2E_SDMT_PASSWORD=SecurePassword123!
# EXEC_RO Role E2E_EXEC_EMAIL=exec-user@example.com E2E_EXEC_PASSWORD=SecurePassword
# NO_GROUP User (CRITICAL for security testing) E2E_NO_GROUP_EMAIL=no-group-
user@example.com E2E_NO_GROUP_PASSWORD=SecurePassword123!
```

⚠ **Important:** The NO_GROUP user must exist in Cognito but **must not** be assigned to any groups. This validates that users without groups are properly denied access.

1.4.2 Running API Tests

```
[ ] # Run all behavioral API tests npm run test:behavioral
# Run specific test suite npx tsx --test tests/behavioral-api/health.test.ts npx tsx --
test tests/behavioral-api/cors.test.ts npx tsx --test tests/behavioral-api/rbac.test.ts npx
tsx --test tests/behavioral-api/schema.test.ts
```

1.4.3 Running UI Tests

```
[ ] # Install Playwright browsers (first time only) npx playwright install chromium
# Run all UI behavioral tests npm run test:ui
# Run in headed mode (see browser) npx playwright test tests/e2e/behavioral --headed
# Run specific test file npx playwright test tests/e2e/behavioral/routing-rbac.spec.ts
# View HTML report npx playwright show-report
```

1.5 CI/CD Integration

1.5.1 Behavioral API Tests (Blocking)

Workflow: .github/workflows/behavioral-api-tests.yml

Triggers: - Pull requests to main/develop - Push to main/develop - Manual workflow dispatch

Required Secrets (configure in GitHub repo settings): - E2E_PMO_EMAIL / E2E_PMO_PASSWORD - E2E_SDMT_EMAIL / E2E_SDMT_PASSWORD - E2E_EXEC_EMAIL / E2E_EXEC_PASSWORD - E2E_NO_GROUP_EMAIL / E2E_NO_GROUP_PASSWORD (CRITICAL)

Evidence: Test output prints endpoint name, HTTP status, role, and pass/fail for each test.

1.5.2 UI Playwright Tests (Non-blocking)

Workflow: .github/workflows/ui-playwright.yml

Triggers: - Manual workflow dispatch - Nightly schedule (2 AM UTC)

Artifacts: - Playwright HTML report - Screenshots/videos of failures - Test results JSON

1.6 Configuring Test Users in Cognito

1.6.1 Automated Setup (Recommended)

Use the provided script to create all test users at once:

```
[ ] # Ensure AWS credentials are configured (via aws configure, SSO, or env vars)
./scripts/cognito/setup-test-users.sh
```

This script will: - Create users for PMO, SDMT, EXEC_RO, and NO_GROUP roles - Set passwords to SecureTestPass2025! - Assign appropriate Cognito groups - Verify NO_GROUP user has zero group memberships

After running the script, add the credentials to .env.local or GitHub Actions secrets as shown in the script output.

1.6.2 Manual Setup

Alternatively, create test users manually:

Creating Test Users For each role, create a user in Cognito:

```
[ ] aws cognito-idp admin-create-user \ --user-pool-id us-east-2_FyHLtOhiY \ --username
"pmo-test@example.com" \ --user-attributes Name=email,Value="pmo-test@example.com" \
\ --temporary-password "TempPassword123!" \ --region us-east-2
# Set permanent password aws cognito-idp admin-set-user-password \ --user-pool-
id us-east-2_FyHLtOhiY \ --username "pmo-test@example.com" \ --password "SecurePass-
word123!" \ --permanent \ --region us-east-2
```

1.6.3 Assigning Groups

Assign users to appropriate Cognito groups:

```
[ ] # PMO user aws cognito-idp admin-add-user-to-group \ --user-pool-id us-east-
2_FyHLtOhiY \ --username "pmo-test@example.com" \ --group-name PMO \ --region
us-east-2
# SDMT user (add to SDT, FIN, or AUD group) aws cognito-idp admin-add-user-to-
group \ --user-pool-id us-east-2_FyHLtOhiY \ --username "sdmt-test@example.com" \
--group-name SDT \ --region us-east-2
# EXEC_RO user aws cognito-idp admin-add-user-to-group \ --user-pool-id us-east-
2_FyHLtOhiY \ --username "exec-test@example.com" \ --group-name EXEC_RO \ --region
us-east-2
# NO_GROUP user - DO NOT add to any group! # This user must have zero group
memberships
```

1.6.4 Verifying Group Membership

```
[ ] aws cognito-idp admin-list-groups-for-user \ --user-pool-id us-east-2_FyHLtOhiY \
--username "no-group-test@example.com" \ --region us-east-2
# Should return empty Groups array: # { # "Groups": [ ] # }
```

1.7 Security: NO_GROUP User Validation

The most critical security test is the NO_GROUP user validation:

1.7.1 Why It's Important

Previously, users with no Cognito groups were silently granted EXEC_RO role by default. This is a **security vulnerability** because: - Any authenticated user could access protected data - Role-based access control was effectively bypassed - Users could see data they shouldn't have access to

1.7.2 The Fix

The security fix ensures: 1. `mapGroupsToRoles()` returns **empty array** for users with no groups 2. `getAvailableRoles()` returns **empty array** (no EXEC_RO fallback) 3. UI shows "no access" message for users with empty roles 4. API backend denies access (401/403) for users with no valid groups

1.7.3 Test Validation

The behavioral tests validate:

```
[ ] // API Test const credentials = getRoleCredentials("NO_GROUP"); const token
= await getCognitoToken(credentials); const result = await apiRequest("/projects",
token);
//MUST return 401 or 403, NEVER 200 assert.ok(result.status === 401 || result.status
=== 403);
//UI Test await setupAuthenticatedPage(page, credentials); await page.goto("/projects");
//MUST redirect to login or show access denied const isBlocked = url.includes("/login")
|| content.includes("access denied"); assert.ok(isBlocked);
```

1.8 Troubleshooting

1.8.1 Common Issues

"Authentication failed" errors **Cause:** Invalid credentials or user not confirmed
Solution: - Verify credentials in `.env.local` - Confirm user in Cognito console - Check user is not in "FORCE_CHANGE_PASSWORD" status

CORS preflight failures **Cause:** CloudFront origin not configured in API **Solution:**
 - Verify API Gateway CORS settings - Check CloudFront origin is in allowed origins list - Ensure headers include authorization and content-type

NO_GROUP tests fail with 200 status **Cause:** Security vulnerability - user granted access without groups **Solution:** - Verify `mapGroupsToRoles()` returns empty array -

Check `getAvailableRoles()` has no `EXEC_RO` fallback - Review backend RBAC implementation

Tests skip due to missing credentials **Cause:** Role credentials not configured

Solution: Add `E2E_*_EMAIL` and `E2E_*_PASSWORD` to environment

1.9 Best Practices

1. **Keep NO_GROUP User Clean:** Never add the `NO_GROUP` test user to any Cognito groups
2. **Test with Real Data:** Don't rely on seeded/canonical data - tests should work with production data
3. **Validate Both API and UI:** Backend and frontend RBAC should match
4. **Monitor Security Tests:** `NO_GROUP` tests are critical - never skip them
5. **Evidence Logging:** Always print test evidence (endpoint, status, role, result)

1.10 Related Documentation

- Authentication Flow
- Auth Validation Guide
- API Contract Tests
- Playwright Configuration