Playwright E2E Testing Guide for CareLinkAI RBAC System

Overview

This guide provides comprehensive documentation for the Playwright E2E test suite that validates the Role-Based Access Control (RBAC) system in CareLinkAI.

Test Structure

```
tests/
├── helpers/
│   └── auth.ts              # Authentication utilities
├── fixtures/
│   └── test-data.ts         # Test data and selectors
├── auth.spec.ts             # Authentication tests
├── residents.spec.ts        # Residents CRUD tests
├── assessments.spec.ts      # Assessments tab tests
├── incidents.spec.ts        # Incidents tab tests
├── compliance.spec.ts       # Compliance tab tests (restricted)
├── family.spec.ts           # Family contacts tests
├── navigation.spec.ts       # Navigation visibility tests
└── dashboard.spec.ts        # Dashboard actions tests
```

Test Users

Prerequisites

Before running tests, you must seed the database with test users:

```
npm run seed:test-users
```

Or manually compile and run:

```
npx tsc --skipLibCheck prisma/seed-test-users.ts
node prisma/seed-test-users.js
```

Test User Credentials

Role	Email	Password
Admin	admin.test@carelinkai.com	TestPassword123!
Operator	operator.test@carelinkai.com	TestPassword123!
Caregiver	caregiver.test@carelinkai.com	TestPassword123!
Family	family.test@carelinkai.com	TestPassword123!

Test Data Created

The seed script creates:
- Test users for each role
- Test assisted living home (assigned to operator)
- Test resident (assigned to family and operator's home)
- Test caregiver employment relationship
- Sample assessment, incident, and compliance items

Running Tests

All Tests

```
npm run test:e2e
```

Specific Test File

```
npx playwright test tests/auth.spec.ts
```

Headed Mode (See Browser)

```
npm run test:e2e:headed
```

Debug Mode

```
npm run test:e2e:debug
```

UI Mode (Interactive)

```
npm run test:e2e:ui
```

Generate and View Reports

```
npm run test:e2e:report
```

Test Coverage

1. Authentication Tests (`auth.spec.ts`)

Scenarios:
- ✅ Display login page
- ✅ Reject invalid credentials
- ✅ Login as Admin, Operator, Caregiver, Family
- ✅ Maintain session on page reload
- ✅ Logout successfully
- ✅ Redirect to login when accessing protected routes
- ✅ Verify role-specific access after login

Expected Results:
- All roles can successfully authenticate
- Invalid credentials are rejected
- Sessions persist across page reloads
- Unauthorized access is blocked

2. Residents Module Tests (`residents.spec.ts`)

Admin Access

- ✅ View all residents
- ✅ Create new resident
- ✅ Edit resident
- ✅ Delete resident
- ✅ All action buttons visible

Operator Access

- ✅ View residents in their homes
- ✅ Create resident in their homes
- ✅ Edit resident in their homes
- ✅ Delete resident in their homes
- ✅ Cannot see other operators' residents

Caregiver Access

- ✅ View assigned residents
- ❌ Cannot create residents
- ❌ Cannot edit residents
- ❌ Cannot delete residents
- ✅ Edit/delete buttons hidden

Family Access

- ✅ View only their family member
- ❌ Cannot create residents
- ❌ Cannot edit residents
- ❌ Cannot delete residents
- ✅ "View Only" badge visible
- ✅ All action buttons hidden
- ❌ Cannot access other residents

3. Assessments Tab Tests (`assessments.spec.ts`)

Admin/Operator (Full Access)

- ✅ View assessments tab
- ✅ Create assessment
- ✅ Delete assessment

Caregiver (Limited)

- ✅ View assessments
- ✅ Create assessments
- ⚠️ Limited delete permissions

Family (View Only)

- ✅ View assessments
- ❌ Cannot create assessments
- ❌ Cannot delete assessments
- ✅ No action buttons visible

4. Incidents Tab Tests (`incidents.spec.ts`)

Admin/Operator (Full Access)

- ✅ View incidents tab
- ✅ Create incident
- ✅ Resolve incident

Caregiver

- ✅ View incidents
- ✅ Report incidents
- ⚠️ Limited resolve permissions

Family (View Only)

- ✅ View incidents
- ❌ Cannot report incidents
- ❌ Cannot resolve incidents
- ✅ No action buttons visible

5. Compliance Tab Tests (`compliance.spec.ts`)

Admin/Operator Only

- ✅ View compliance tab
- ✅ Manage compliance items
- ✅ No restricted access message

Caregiver (Restricted)

- ❌ “Restricted Access” message shown
- ❌ Cannot see compliance items
- ❌ Cannot add compliance items

Family (Restricted)

- ❌ “Restricted Access” message shown
- ❌ Cannot see compliance items

- ❌ Cannot add compliance items
- ⚠️ Tab may be completely hidden

6. Family Tab Tests (`family.spec.ts`)

Admin/Operator (Full Access)

- ✅ View family contacts
- ✅ Add family contact
- ✅ Edit family contact
- ✅ Delete family contact

Caregiver (View Only)

- ✅ View family contacts
- ❌ Cannot add contacts
- ❌ Cannot edit contacts
- ❌ Cannot delete contacts

Family (View Only)

- ✅ View family contacts
- ❌ Cannot add contacts
- ❌ Cannot edit contacts
- ❌ Cannot delete contacts
- ✅ No action buttons visible

7. Navigation Tests (`navigation.spec.ts`)

Admin

- ✅ Sees all menu items
- ✅ Can access Operators page
- ✅ Sees Admin Tools

Operator

- ✅ Sees operator menu items
- ❌ Cannot see Operators menu (admin-only)
- ❌ Cannot access admin pages
- ❌ Does not see Admin Tools

Caregiver

- ✅ Sees limited menu items
- ❌ Cannot see Operators menu
- ❌ Cannot see Admin Tools
- ✅ Minimal navigation options

Family

- ✅ Sees minimal menu items
- ❌ Cannot see Operators menu
- ❌ Cannot see Caregivers menu
- ❌ Cannot see Admin Tools
- ✅ Very limited navigation

8. Dashboard Tests (`dashboard.spec.ts`)

Admin

- ✅ Can access dashboard
- ✅ Sees all quick action buttons
- ✅ Sees system-wide KPIs
- ✅ No data restrictions

Operator

- ✅ Can access dashboard
- ✅ Sees scoped quick actions
- ✅ Sees KPIs for their homes only
- ❌ Cannot see system-wide metrics

Caregiver

- ✅ Can access dashboard
- ✅ Sees their schedule and tasks
- ❌ No management buttons
- ❌ Cannot access management features

Family

- ✅ Can access dashboard
- ✅ Sees resident information
- ❌ No management actions
- ✅ Dashboard is view-only
- ⚠️ May see “View Only” indicator

Configuration

Playwright Config (`playwright.config.ts`)

```
{
  testDir: './tests',
  timeout: 30 * 1000,
  fullyParallel: false, // Avoid race conditions
  retries: process.env.CI ? 2 : 0,
  reporter: ['html', 'json', 'list'],
  use: {
    baseURL: 'http://localhost:3000',
    trace: 'on-first-retry',
    screenshot: 'only-on-failure',
    video: 'retain-on-failure',
  },
  webServer: {
    command: 'npm run dev',
    url: 'http://localhost:3000',
    reuseExistingServer: !process.env.CI,
  },
}
```

Environment Variables

Set `PLAYWRIGHT_BROWSERS_PATH` if needed:

```
export PLAYWRIGHT_BROWSERS_PATH=$HOME/.cache/ms-playwright
```

Troubleshooting

Database Connection Issues

If you see “Can’t reach database server”:

1. Ensure PostgreSQL is running
2. Check `DATABASE_URL` in `.env`
3. Run migrations: `npm run migrate:deploy`

Test Users Not Found

If tests fail with “Invalid credentials”:

1. Run seed script: `npm run seed:test-users`
2. Verify users exist in database
3. Check credentials in `tests/helpers/auth.ts`

Timeout Errors

If tests timeout frequently:

1. Increase timeout in `playwright.config.ts`
2. Check server is running: `npm run dev`
3. Verify network connectivity

Element Not Found

If tests fail with “Element not found”:

1. Check selectors in `tests/fixtures/test-data.ts`
2. Update selectors to match current UI
3. Use `test:e2e:headed` to debug visually

Best Practices

Writing Tests

1. **Use Helpers**: Import auth and data helpers
2. **Wait for Ready**: Call `waitForPageReady()` after navigation
3. **Flexible Selectors**: Use multiple selector strategies
4. **Avoid Hard Waits**: Use Playwright’s auto-waiting
5. **Clean Test Data**: Don’t actually delete test data

Maintaining Tests

1. **Update Selectors**: When UI changes, update `test-data.ts`
2. **Add New Tests**: For new features, add corresponding tests
3. **Keep Isolated**: Each test should be independent
4. **Document Changes**: Update this guide when adding tests

CI/CD Integration

GitHub Actions Example

```yaml
name: E2E Tests

on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
      - run: npm ci
      - run: npx playwright install --with-deps
      - run: npm run seed:test-users
      - run: npm run test:e2e
      - uses: actions/upload-artifact@v3
        if: always()
        with:
          name: playwright-report
          path: playwright-report/
```

Test Results

After running tests, view the HTML report:

```
npm run test:e2e:report
```

Reports include:
- ✅ Passed tests
- ❌ Failed tests
- ⏭️ Skipped tests
- 📸 Screenshots on failure
- 🎥 Videos on failure
- 📊 Trace files for debugging

Next Steps

1. **Run Tests Locally**: Validate RBAC implementation
2. **Fix Failing Tests**: Address any permission issues
3. **Add More Tests**: Expand coverage as needed
4. **Integrate CI/CD**: Add to deployment pipeline
5. **Monitor Results**: Track test health over time

Support

For questions or issues:
- Review Playwright docs: https://playwright.dev
- Check CareLinkAI RBAC docs: `PHASE_4_RBAC_IMPLEMENTATION.md`
- Contact development team

Last Updated: December 2024
Version: 1.0.0
Status: ✅ Complete and Ready for Testing