JSUI Testing Tools Sketch

Overview

This document outlines the tools and approach for automated testing of both the User and Admin SPAs.

---

1. Testing Framework Selection

1.1 Recommended Stack

Tool	Purpose
Playwright	Browser automation, E2E testing
Vitest	Unit testing (ES modules native)
MSW (Mock Service Worker)	API mocking for isolated tests

1.2 Rationale

- **Playwright** over Cypress: Better ES module support, cross-browser testing, no server required
- **Vitest** over Jest: Native ES module support matches the JSUI's vanilla ES6 architecture
- **MSW**: Intercepts fetch/XHR at network level; works with both Vitest and Playwright

---

2. Test Categories

2.1 Unit Tests (Vitest)

Test individual modules in isolation:

```
tests/unit/
  lib/
      auth.test.js           # authenticateAccount, token handling
      auth_store.test.js     # Token storage/retrieval
      app_state.test.js      # State management
      models/
          account.test.js
          repository.test.js
          session.test.js
  ui/
      utils.test.js          # escapeHtml, joinHtml
      router.test.js         # Route matching
```

```
  js/
      sparql-editor.test.js  # Editor utilities
```

2.2 Component Tests (Vitest + jsdom)

Test page classes with mocked DOM:

```
tests/components/
  pages/
      LoginPage.test.js
      RepositoryShowPage.test.js
      SparqlPage.test.js
  admin/
      AdminLoginPage.test.js
      AdminDashboardPage.test.js
```

2.3 Integration Tests (Playwright)

Test full user flows with mocked backend:

```
tests/e2e/
  user/
      login.spec.js
      account-management.spec.js
      repository-crud.spec.js
      import-export.spec.js
      sparql-query.spec.js
      view-management.spec.js
  admin/
      admin-login.spec.js
      account-management.spec.js
      repository-management.spec.js
      invitation-management.spec.js
      query-history.spec.js
      transaction-history.spec.js
      import-history.spec.js
```

2.4 API Contract Tests

Verify API request/response format expectations:

```
tests/api/
  auth-endpoints.test.js
  account-endpoints.test.js
  repository-endpoints.test.js
  history-endpoints.test.js
  graph-store-endpoints.test.js
```

---

3. Mock Server Setup

3.1 MSW Handlers

```
// tests/mocks/handlers.js
import { http, HttpResponse } from 'msw';

export const handlers = [
  // Authentication
  http.post('/system/accounts/:acct/configuration', ({ params }) => {
    return HttpResponse.json({
      token: 'test-token-12345',
      configuration: { name: params.acct, email: 'test@example.com' }
    });
  }),

  // Admin privilege check
  http.get('/system/users/:acct/configuration', ({ params }) => {
    return HttpResponse.json({
      administrator_of: params.acct === 'admin' ? 'http://dydra.com' : null
    });
  }),

  // Account list
  http.get('/system/accounts', () => {
    return HttpResponse.json(['alice', 'bob', 'charlie']);
  }),

  // Repository list
  http.get('/system/accounts/:acct/repositories', () => {
    return HttpResponse.json(['repo1', 'repo2']);
  }),

  // Query history
  http.get('/system/accounts/:acct/repositories/:repo/service_history', () => {
    return HttpResponse.json([
      { timestamp: '2026-02-05T10:00:00Z', elapsed_time: 150, signature: 'abc123' }
    ]);
  }),

  // Transaction history
  http.get('/system/service_history/transactions', () => {
    return HttpResponse.json({ data: [] });
  }),

  // Import history
```

```
  http.get('/system/service_history/imports', () => {
    return HttpResponse.json({ data: [] });
  }),

  // Query text
  http.get('/system/service_history/queries/:acct/:sig', () => {
    return new HttpResponse('SELECT * WHERE { ?s ?p ?o }', {
      headers: { 'Content-Type': 'application/sparql-query' }
    });
  }),
];
```

3.2 Test Fixtures

```
// tests/fixtures/accounts.js
export const testAccounts = {
  admin: {
    name: 'admin',
    email: 'admin@dydra.com',
    administrator_of: 'http://dydra.com'
  },
  user: {
    name: 'testuser',
    email: 'user@example.com',
    administrator_of: null
  }
};

// tests/fixtures/repositories.js
export const testRepositories = {
  foaf: {
    name: 'foaf',
    quad_count: 1000,
    summary: 'FOAF data'
  }
};
```

---

4. E2E Test Examples

4.1 User App: Login Flow

```
// tests/e2e/user/login.spec.js
import { test, expect } from '@playwright/test';

test.describe('User Login', () => {
```

```
  test('successful login redirects to home', async ({ page }) => {
    await page.goto('/login');

    await page.fill('input[name="login"]', 'testuser');
    await page.fill('input[name="password"]', 'testpass');
    await page.click('input[type="submit"]');

    await expect(page).toHaveURL('/');
    await expect(page.locator('.user-name')).toContainText('testuser');
  });

  test('invalid credentials show error', async ({ page }) => {
    await page.goto('/login');

    await page.fill('input[name="login"]', 'baduser');
    await page.fill('input[name="password"]', 'badpass');
    await page.click('input[type="submit"]');

    await expect(page.locator('#login-error')).toBeVisible();
  });

  test('invite link is visible on login page', async ({ page }) => {
    await page.goto('/login');

    await expect(page.locator('a[href*="/invite"]')).toContainText('Request an invitation');
  });
});
```

4.2 User App: Repository Import

```
// tests/e2e/user/import-export.spec.js
import { test, expect } from '@playwright/test';
import path from 'path';

test.describe('Repository Import', () => {
  test.beforeEach(async ({ page }) => {
    // Login first
    await page.goto('/login');
    await page.fill('input[name="login"]', 'testuser');
    await page.fill('input[name="password"]', 'testpass');
    await page.click('input[type="submit"]');
    await page.waitForURL('/');
  });

  test('import shows progress and can be cancelled', async ({ page }) => {
    await page.goto('/account/testuser/repositories/testrepo');
```

```
    // Start import
    const fileChooserPromise = page.waitForEvent('filechooser');
    await page.click('.repo-import-btn');
    const fileChooser = await fileChooserPromise;
    await fileChooser.setFiles(path.join(__dirname, 'fixtures/test.ttl'));

    // Confirm media type popup
    await page.click('button:has-text("Import")');

    // Check cancel button appears
    await expect(page.locator('.repo-import-btn')).toContainText('Cancel');

    // Export button should be disabled
    await expect(page.locator('.repo-export-btn')).toBeDisabled();
  });

  test('export downloads file with correct format', async ({ page }) => {
    await page.goto('/account/testuser/repositories/testrepo');

    await page.click('.repo-export-btn');
    await page.selectOption('select', 'text/turtle');

    const downloadPromise = page.waitForEvent('download');
    await page.click('button:has-text("Export")');
    const download = await downloadPromise;

    expect(download.suggestedFilename()).toMatch(/\.ttl$/);
  });
});
```

4.3 Admin App: Authorization Model

```
// tests/e2e/admin/authorization.spec.js
import { test, expect } from '@playwright/test';

test.describe('Admin Authorization', () => {
  test('admin user sees all controls', async ({ page }) => {
    await page.goto('/admin/login');
    await page.fill('input[name="login"]', 'admin');
    await page.fill('input[name="password"]', 'adminpass');
    await page.click('input[type="submit"]');

    await page.waitForURL('/admin/');

    // Accounts tab
```

```
    await expect(page.locator('.admin-new-account-btn')).toBeVisible();
    await expect(page.locator('.account-delete-btn').first()).toBeVisible();
  });

  test('non-admin user sees data but not destructive controls', async ({ page }) => {
    await page.goto('/admin/login');
    await page.fill('input[name="login"]', 'regularuser');
    await page.fill('input[name="password"]', 'userpass');
    await page.click('input[type="submit"]');

    await page.waitForURL('/admin/');

    // Data is visible
    await expect(page.locator('table.admin')).toBeVisible();

    // Admin controls are hidden
    await expect(page.locator('.admin-new-account-btn')).not.toBeVisible();
    await expect(page.locator('.account-delete-btn')).not.toBeVisible();
  });
});
```

4.4 Admin App: Query History Refresh

```
// tests/e2e/admin/query-history.spec.js
import { test, expect } from '@playwright/test';

test.describe('Query History', () => {
  test('refreshes when repository selection changes', async ({ page }) => {
    // Login as admin
    await page.goto('/admin/login');
    await page.fill('input[name="login"]', 'admin');
    await page.fill('input[name="password"]', 'adminpass');
    await page.click('input[type="submit"]');

    // Select an account
    await page.click('.account-checkbox[data-account="testuser"]');

    // Go to repositories tab and select repos
    await page.click('a[href="#admin-repositories"]');
    await page.click('.repo-checkbox[data-repo="repo1"]');

    // Go to query history tab
    await page.click('a[href="#admin-query-history"]');
    await expect(page.locator('#admin-query-history table')).toBeVisible();

    // Change repo selection
```

```
    await page.click('a[href="#admin-repositories"]');
    await page.click('.repo-checkbox[data-repo="repo2"]');

    // Return to query history - should refresh
    await page.click('a[href="#admin-query-history"]');
    // Content should have been refreshed (verify via request count or content)
  });

  test('signature links open query text dialog', async ({ page }) => {
    // ... login and navigate to query history with data ...

    await page.click('.query-signature-link');

    await expect(page.locator('pre')).toContainText('SELECT');
    await page.click('button:has-text("OK")');
  });
});
```

---

5. Configuration Files

5.1 Vitest Configuration

```
// vitest.config.js
import { defineConfig } from 'vitest/config';

export default defineConfig({
  test: {
    environment: 'jsdom',
    include: ['tests/unit/**/*.test.js', 'tests/components/**/*.test.js'],
    setupFiles: ['tests/setup.js'],
  },
});
```

5.2 Playwright Configuration

```
// playwright.config.js
import { defineConfig } from '@playwright/test';

export default defineConfig({
  testDir: './tests/e2e',
  use: {
    baseURL: 'http://localhost:8080',
  },
  webServer: {
    command: 'npx serve -l 8080',
```

```
    port: 8080,
    reuseExistingServer: !process.env.CI,
  },
});
```

5.3 Test Setup

```
// tests/setup.js
import { setupServer } from 'msw/node';
import { handlers } from './mocks/handlers.js';

export const server = setupServer(...handlers);

beforeAll(() => server.listen());
afterEach(() => server.resetHandlers());
afterAll(() => server.close());
```

---

6. CI Integration

6.1 GitHub Actions Workflow

```
# .github/workflows/test.yml
name: Test

on: [push, pull_request]

jobs:
  unit-tests:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm ci
      - run: npm run test:unit

  e2e-tests:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm ci
```

```
      - run: npx playwright install --with-deps
      - run: npm run test:e2e
```

---

7. Test Coverage Goals

Area	Target
Unit tests (lib/)	90%
Component tests (pages/)	70%
E2E flows (critical paths)	100%
API contract tests	100%

7.1 Critical Paths for E2E

User App: 1. Login → View repositories → Open repository → Execute SPARQL query 2. Login → Import file → Verify success message 3. Login → Export repository → Verify download 4. Login → Create/delete view

Admin App: 1. Admin login → View accounts → Create account → Delete account 2. Admin login → Select accounts → View repositories → Delete repository 3. Admin login → View invitations → Send invitation → Delete invitation 4. Admin login → Select repositories → View query history → Click signature link 5. Non-admin login → Verify hidden controls

---

8. Package.json Scripts

```
{
  "scripts": {
    "test": "npm run test:unit && npm run test:e2e",
    "test:unit": "vitest run",
    "test:unit:watch": "vitest",
    "test:e2e": "playwright test",
    "test:e2e:ui": "playwright test --ui",
    "test:coverage": "vitest run --coverage"
  },
  "devDependencies": {
    "vitest": "^1.0.0",
    "@vitest/coverage-v8": "^1.0.0",
    "jsdom": "^24.0.0",
    "msw": "^2.0.0",
    "@playwright/test": "^1.40.0"
  }
}
```