

# Requirements Document

## Application Overview

The Dydra JavaScript User Interface (JSUI) is a single-page web application (SPA) that provides a comprehensive interface for managing RDF data repositories, executing SPARQL queries, and administering accounts on the Dydra platform. The application is built using vanilla JavaScript with a component-based architecture and client-side routing.

## Functional Requirements

### 1. User Authentication and Session Management

#### 1.1 Authentication

- **REQ-1.1.1:** The system shall support user authentication via Basic Authentication (username/password) or Bearer token authentication.
- **REQ-1.1.2:** The system shall authenticate users against the Dydra service API endpoint `/system/accounts/<account>/configuration`.
- **REQ-1.1.3:** Upon successful authentication, the system shall store the access token for subsequent API requests.
- **REQ-1.1.4:** The system shall support multiple authenticated accounts simultaneously.

#### 1.2 Session Management

- **REQ-1.2.1:** The system shall maintain session state across page reloads using browser storage.
- **REQ-1.2.2:** The system shall provide logout functionality that clears session data and redirects to the login page.
- **REQ-1.2.3:** The system shall display different navigation links based on authentication status (logged in vs. logged out).

### 2. Account Management

#### 2.1 Account Display

- **REQ-2.1.1:** The system shall display account information including friendly ID, email, full name, homepage, blog, company, and location.
- **REQ-2.1.2:** The system shall support viewing account details for the current authenticated account.
- **REQ-2.1.3:** The system shall support viewing account details for other accounts by account name.

#### 2.2 Account Editing

- **REQ-2.2.1:** The system shall support editing account profile fields (first-name, familyname, fullname, email, homepage, blog, company, location, phone, skype\_id, jabber\_id, workinfo).
- **REQ-2.2.2:** The system shall track only edited fields and send only modified data to the server.
- **REQ-2.2.3:** The system shall support saving account configuration changes via POST to `/system/accounts/<account>/configuration`.
- **REQ-2.2.4:** The system shall display authentication tokens for accounts.

## 2.3 Account Creation

- **REQ-2.3.1:** The system shall provide functionality to create new accounts (subject to invitation requirements if configured).

## 3. Repository Management

### 3.1 Repository Display

- **REQ-3.1.1:** The system shall display a list of repositories for a given account.
- **REQ-3.1.2:** The system shall display repository details including name, summary, description, homepage, quad count, disk size, license, and import status.
- **REQ-3.1.3:** The system shall support viewing repository details by account name and repository name.

### 3.2 Repository Editing

- **REQ-3.2.1:** The system shall support editing repository metadata fields (homepage, summary, description, abstract, privacy\_setting, permissible\_ip\_addresses, prefixes).
- **REQ-3.2.2:** The system shall track only edited fields and send only modified data to the server.
- **REQ-3.2.3:** The system shall support saving repository configuration changes via POST to `/system/accounts/<account>/repositories/<repository>/configuration`.

### 3.3 Repository Creation

- **REQ-3.3.1:** The system shall provide functionality to create new repositories within an account.

### 3.4 Repository Import

- **REQ-3.4.1:** The system shall support importing data into repositories.
- **REQ-3.4.2:** The system shall display import job status and success/failure notifications.

## 4. SPARQL Query Management

### 4.1 Query Execution

- **REQ-4.1.1:** The system shall provide a SPARQL query editor with syntax highlighting and autocomplete capabilities.
- **REQ-4.1.2:** The system shall support executing SPARQL queries against repository endpoints.
- **REQ-4.1.3:** The system shall support multiple response media types:
  - JSON (application/sparql-results+json)
  - XML (application/sparql-results+xml)
  - SVG (image/vnd.dydra.SPARQL-RESULTS+GRAPHVIZ+SVG+XML)
  - HTML (text/html)
  - CSV (text/csv)
  - TSV (text/tab-separated-values)
  - Turtle (text/turtle)
  - N-Triples (application/n-triples)
  - RDF/XML (application/rdf+xml)
  - JSON-LD (application/ld+json)
  - SSE (application/sparql-query+sse)
- **REQ-4.1.4:** The system shall display query results in appropriate formats (tables for JSON, formatted text for XML/RDF, rendered content for HTML/SVG).
- **REQ-4.1.5:** The system shall support parameterized queries with dynamic input fields.
- **REQ-4.1.6:** The system shall maintain a history of executed queries with timestamps and execution times.

### 4.2 View Management

- **REQ-4.2.1:** The system shall support creating, reading, updating, and deleting SPARQL views (saved queries).
- **REQ-4.2.2:** The system shall load view query text from `/system/accounts/<account>/repositories/`
- **REQ-4.2.3:** The system shall save view query text via PUT to the view endpoint.
- **REQ-4.2.4:** The system shall list views from the repository configuration response.

### 4.3 Query Editor Features

- **REQ-4.3.1:** The system shall provide a collapsible query editor interface.
- **REQ-4.3.2:** The system shall support keyboard shortcuts (Ctrl/Cmd+Enter to execute).
- **REQ-4.3.3:** The system shall provide query reset functionality to restore original query text.
- **REQ-4.3.4:** The system shall display execution time and result counts for queries.

- **REQ-4.3.5:** The system shall provide an event log for query execution activities.

## 5. Navigation and Routing

### 5.1 Client-Side Routing

- **REQ-5.1.1:** The system shall implement client-side routing using the History API or hash-based routing.
- **REQ-5.1.2:** The system shall support browser back/forward navigation.
- **REQ-5.1.3:** The system shall support programmatic navigation.
- **REQ-5.1.4:** The system shall handle route parameters (e.g., account names, repository names, view names).
- **REQ-5.1.5:** The system shall support a configurable base path for deployment in subdirectories.

### 5.2 Navigation UI

- **REQ-5.2.1:** The system shall provide navigation links for Home, About, Docs, and Blog (external links).
- **REQ-5.2.2:** The system shall provide navigation links for My Account and Logout when authenticated.
- **REQ-5.2.3:** The system shall provide navigation links for Signup and Login when not authenticated.
- **REQ-5.2.4:** The system shall provide a location bar showing the current route path.
- **REQ-5.2.5:** The system shall support tabbed interfaces for managing multiple open accounts and repositories.

## 6. User Interface Components

### 6.1 Layout

- **REQ-6.1.1:** The system shall provide a consistent page layout with header, navigation, main content area, optional sidebar, and footer.
- **REQ-6.1.2:** The system shall support a home layout variant for landing pages.
- **REQ-6.1.3:** The system shall support responsive design for different screen sizes.

### 6.2 Page Components

- **REQ-6.2.1:** The system shall provide flash message components for user notifications.
- **REQ-6.2.2:** The system shall provide loading indicators for asynchronous operations.
- **REQ-6.2.3:** The system shall provide form components with validation feedback.

### 6.3 Tab Management

- **REQ-6.3.1:** The system shall support tabbed interfaces for organizing content.
- **REQ-6.3.2:** The system shall support closing tabs and managing tab state.
- **REQ-6.3.3:** The system shall track open accounts and repositories in tabs.

## 7. Data Persistence and Synchronization

### 7.1 Data Replication

- **REQ-7.1.1:** The system shall implement a replication system for synchronizing local state with remote server state.
- **REQ-7.1.2:** The system shall track object state changes (new, clean, modified, deleted) using a JDO/JPA-like state machine.
- **REQ-7.1.3:** The system shall generate delta maps for tracking property changes.
- **REQ-7.1.4:** The system shall support rollback of changes to restore previous state.

### 7.2 Persistence Adapter

- **REQ-7.2.1:** The system shall provide an abstraction layer for data persistence operations.
- **REQ-7.2.2:** The system shall support RDF-based storage backends via an adapter interface.
- **REQ-7.2.3:** The system shall support fallback to cached/sample data when persistence is unavailable.

## 8. Invitation System

### 8.1 Invitation Management

- **REQ-8.1.1:** The system shall support creating invitations for new user signups (if invitation requirement is enabled).
- **REQ-8.1.2:** The system shall display a list of invitations.
- **REQ-8.1.3:** The system shall show invitation success confirmation.

## 9. Error Handling

### 9.1 Error Display

- **REQ-9.1.1:** The system shall display user-friendly error messages for authentication failures.
- **REQ-9.1.2:** The system shall display error messages for API request failures.
- **REQ-9.1.3:** The system shall display a 404 page for unknown routes.

## 9.2 Error Recovery

- **REQ-9.2.1:** The system shall handle network errors gracefully.
- **REQ-9.2.2:** The system shall provide retry mechanisms for failed operations where appropriate.

## 10. Configuration

### 10.1 Application Configuration

- **REQ-10.1.1:** The system shall support configurable base host, docs host, and blog host.
- **REQ-10.1.2:** The system shall support configurable base path for deployment flexibility.
- **REQ-10.1.3:** The system shall support optional invitation requirement for signups.
- **REQ-10.1.4:** The system shall support optional display of account balances.

## Non-Functional Requirements

### 11. Performance

#### 11.1 Response Time

- **REQ-11.1.1:** The system shall render initial page load within 2 seconds on standard broadband connections.
- **REQ-11.1.2:** The system shall update UI in response to user actions within 100ms for local operations.

#### 11.2 Resource Usage

- **REQ-11.2.1:** The system shall minimize memory usage by cleaning up unused event listeners and DOM references.
- **REQ-11.2.2:** The system shall limit event log entries to prevent memory issues (default: 100 entries).

### 12. Browser Compatibility

#### 12.1 Supported Browsers

- **REQ-12.1.1:** The system shall support modern browsers with ES6+ JavaScript support.
- **REQ-12.1.2:** The system shall support browsers with History API or fallback to hash-based routing.

### 13. Security

#### 13.1 Authentication

- **REQ-13.1.1:** The system shall securely store authentication tokens (preferably in memory, with optional secure storage).
- **REQ-13.1.2:** The system shall not expose passwords in logs or error messages.

## 13.2 Data Protection

- **REQ-13.2.1:** The system shall use HTTPS for all API communications.
- **REQ-13.2.2:** The system shall validate and sanitize user input before sending to the server.

## 14. Maintainability

### 14.1 Code Organization

- **REQ-14.1.1:** The system shall organize code into logical modules (routing, UI components, models, persistence).
- **REQ-14.1.2:** The system shall use consistent naming conventions and code style.

### 14.2 Documentation

- **REQ-14.2.1:** The system shall provide inline code documentation for complex logic.
- **REQ-14.2.2:** The system shall maintain API documentation for external integrations.

## External Dependencies

## 15. External Services

### 15.1 Dydra API

- **REQ-15.1.1:** The system shall integrate with the Dydra service API for all data operations.
- **REQ-15.1.2:** The system shall handle API versioning and backward compatibility.

## 16. External Libraries

### 16.1 UI Libraries

- **REQ-16.1.1:** The system shall use Bootstrap CSS framework for styling.
- **REQ-16.1.2:** The system shall use jQuery UI for certain UI components.
- **REQ-16.1.3:** The system shall use YASQE (Yet Another SPARQL Query Editor) for SPARQL editing.
- **REQ-16.1.4:** The system shall use CodeMirror (via YASQE) for code editing features.

### **Future Enhancements (Out of Scope)**

- Real-time collaboration features
- Advanced query optimization suggestions
- Data visualization beyond basic table display
- Mobile native applications
- Offline mode with local data caching