Requirements Document

Version 2026-02-07

This document supersedes `requirements.md` and reflects the current implementation status as of February 7, 2026. Each requirement is marked with its status: **satisfied**, **to be satisfied**, or **deprecated**.

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.

The application consists of two separate SPAs: - **User App** (`index.html`): General user interface for account management, repository operations, SPARQL queries, and data import/export - **Admin App** (`admin.html`): Administrative interface for system-wide management, account/repository administration, invitation handling, and history monitoring

Both applications share core infrastructure (router, authentication, state management) and communicate with the SPOCQ backend via a RESTful `/system/` API.

Functional Requirements

1. User Authentication and Session Management

1.1 Authentication

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

1.2 Session Management

- **REQ-1.2.1** [satisfied]: The system shall maintain session state across page reloads using browser storage.
- **REQ-1.2.2** [satisfied]: The system shall provide logout functionality that clears session data and redirects to the login page.

- **REQ-1.2.3** [satisfied]: 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** [satisfied]: The system shall display account information including friendly ID, email, full name, homepage, blog, company, and location.
- **REQ-2.1.2** [satisfied]: The system shall support viewing account details for the current authenticated account.
- **REQ-2.1.3** [satisfied]: The system shall support viewing account details for other accounts by account name.

2.2 Account Editing

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

2.3 Account Creation

- **REQ-2.3.1** [satisfied]: 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** [satisfied]: The system shall display a list of repositories for a given account.
- **REQ-3.1.2** [satisfied]: The system shall display repository details including name, summary, description, homepage, quad count, disk size, license, and import status.
- **REQ-3.1.3** [satisfied]: The system shall support viewing repository details by account name and repository name.

3.2 Repository Editing

- **REQ-3.2.1** [satisfied]: The system shall support editing repository metadata fields (homepage, summary, description, abstract, privacy_setting, permissible_ip_addresses, prefixes).
- **REQ-3.2.2** [satisfied]: The system shall track only edited fields and send only modified data to the server.
- **REQ-3.2.3** [satisfied]: The system shall support saving repository configuration changes via POST to `/system/accounts/<account>/repositories/<repository>/configurat`

3.3 Repository Creation

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

3.4 Repository Import

- **REQ-3.4.1** [satisfied]: The system shall support importing data into repositories.
- **REQ-3.4.2** [satisfied]: The system shall display import job status and success/failure notifications.
- **REQ-3.4.3** [satisfied]: The system shall support importing data from URLs (web-based imports).
- **REQ-3.4.4** [satisfied]: The system shall support importing data from local file uploads.
- **REQ-3.4.5** [satisfied]: The system shall support asynchronous imports with `AcceptAsynchronous: notify` header.
- **REQ-3.4.6** [satisfied]: The system shall display import progress with byte counts and support abort capability.
- **REQ-3.4.7** [satisfied]: The system shall auto-detect file formats based on file extensions (.ttl, .rdf, .xml, .nt, .nq, .trig, .jsonld, .json, .csv).

3.5 Repository Export

- **REQ-3.5.1** [satisfied]: The system shall support exporting repository data in multiple RDF formats (Turtle, N-Triples, N-Quads, RDF/XML, JSON-LD, TriG, CSV).
- **REQ-3.5.2** [satisfied]: The system shall provide a direct download URL for repository data in Turtle format.

3.6 Repository Operations

- **REQ-3.6.1** [satisfied]: The system shall support clearing all triples from a repository with confirmation dialog.

3.7 Repository Collaboration

- **REQ-3.7.1** [satisfied]: The system shall support managing repository collaborators with read/write permissions.

- **REQ-3.7.2** [satisfied]: The system shall support adding, editing, and removing collaborators.
- **REQ-3.7.3** [satisfied]: The system shall save collaboration changes via POST to `/system/accounts/<account>/repositories/<repository>/collaboration`.

4. SPARQL Query Management

4.1 Query Execution

- **REQ-4.1.1** [satisfied]: The system shall provide a SPARQL query editor with syntax highlighting and autocomplete capabilities.
- **REQ-4.1.2** [satisfied]: The system shall support executing SPARQL queries against repository endpoints.
- **REQ-4.1.3** [satisfied]: 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** [satisfied]: 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** [satisfied]: The system shall support parameterized queries with dynamic input fields.
- **REQ-4.1.6** [satisfied]: The system shall maintain a history of executed queries with timestamps and execution times.

4.2 View Management

- **REQ-4.2.1** [satisfied]: The system shall support creating, reading, updating, and deleting SPARQL views (saved queries).
- **REQ-4.2.2** [satisfied]: The system shall load view query text from `/system/accounts/<account>/repositories/<repository>/views/<view>`.
- **REQ-4.2.3** [satisfied]: The system shall save view query text via PUT to the view endpoint.
- **REQ-4.2.4** [satisfied]: The system shall list views from the repository configuration response.
- **REQ-4.2.5** [satisfied]: The system shall support opening views in the current pane editor.

- **REQ-4.2.6** [satisfied]: The system shall support opening views in a new pane.
- **REQ-4.2.7** [satisfied]: The system shall support opening view results in a new window with table display and multiple output formats.

4.3 Query Editor Features

- **REQ-4.3.1** [satisfied]: The system shall provide a collapsible query editor interface.
- **REQ-4.3.2** [satisfied]: The system shall support keyboard shortcuts (Ctrl/Cmd+Enter to execute).
- **REQ-4.3.3** [satisfied]: The system shall provide query reset functionality to restore original query text.
- **REQ-4.3.4** [satisfied]: The system shall display execution time and result counts for queries.
- **REQ-4.3.5** [satisfied]: The system shall provide an event log for query execution activities.
- **REQ-4.3.6** [satisfied]: The system shall support multiple query tabs within a single editor instance.
- **REQ-4.3.7** [satisfied]: The system shall support dragging query editor tabs outside the application window to open them as standalone windows.
- **REQ-4.3.8** [satisfied]: The system shall preserve query text, query tabs, results, and authentication state when opening a standalone editor window.

5. Navigation and Routing

5.1 Client-Side Routing

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

5.2 Navigation UI

- **REQ-5.2.1** [satisfied]: The system shall provide navigation links for Home, About, Docs, and Blog (external links).
- **REQ-5.2.2** [satisfied]: The system shall provide navigation links for My Account and Logout when authenticated.

- **REQ-5.2.3** [satisfied]: The system shall provide navigation links for Signup and Login when not authenticated.
- **REQ-5.2.4** [satisfied]: The system shall provide a location bar showing the current route path.
- **REQ-5.2.5** [satisfied]: The system shall support tabbed interfaces for managing multiple open accounts and repositories.
- **REQ-5.2.6** [satisfied]: The system shall support cross-app navigation between user and admin SPAs.

6. User Interface Components

6.1 Layout

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

6.2 Page Components

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

6.3 Tab Management

- **REQ-6.3.1** [satisfied]: The system shall support tabbed interfaces for organizing content.
- **REQ-6.3.2** [satisfied]: The system shall support closing tabs and managing tab state.
- **REQ-6.3.3** [satisfied]: The system shall track open accounts and repositories in tabs.
- **REQ-6.3.4** [satisfied]: The system shall support tabbed sub-panes within dashboard panes (e.g., table view and graph view).

7. Data Persistence and Synchronization

7.1 Data Replication

- **REQ-7.1.1** [satisfied]: The system shall implement a replication system for synchronizing local state with remote server state.
- **REQ-7.1.2** [satisfied]: The system shall track object state changes (new, clean, modified, deleted) using a JDO/JPA-like state machine.

- **REQ-7.1.3** [satisfied]: The system shall generate delta maps for tracking property changes.
- **REQ-7.1.4** [satisfied]: The system shall support rollback of changes to restore previous state.

7.2 Persistence Adapter

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

8. Invitation System

8.1 Invitation Management

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

9. Error Handling

9.1 Error Display

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

9.2 Error Recovery

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

10. Configuration

10.1 Application Configuration

- **REQ-10.1.1** [satisfied]: The system shall support configurable base host, docs host, and blog host.
- **REQ-10.1.2** [satisfied]: The system shall support configurable base path for deployment flexibility.

- **REQ-10.1.3** [satisfied]: The system shall support optional invitation requirement for signups.
- **REQ-10.1.4** [satisfied]: The system shall support optional display of account balances.

11. Administration Interface

11.1 Admin Authentication

- **REQ-11.1.1** [satisfied]: The system shall provide a separate admin authentication interface.
- **REQ-11.1.2** [satisfied]: The system shall support role-based access control, showing data to non-admin users but restricting destructive controls.

11.2 Account Administration

- **REQ-11.2.1** [satisfied]: The system shall provide an interface for listing all accounts in the system.
- **REQ-11.2.2** [satisfied]: The system shall support bulk selection of accounts for filtering operations.
- **REQ-11.2.3** [satisfied]: The system shall support viewing account details in a modal dialog.
- **REQ-11.2.4** [satisfied]: The system shall support creating new accounts from the admin interface.
- **REQ-11.2.5** [satisfied]: The system shall support deleting accounts from the admin interface.

11.3 Repository Administration

- **REQ-11.3.1** [satisfied]: The system shall provide an interface for listing repositories filtered by selected accounts.
- **REQ-11.3.2** [satisfied]: The system shall support bulk selection of repositories for filtering operations.

11.4 Admin Invitation Management

- **REQ-11.4.1** [satisfied]: The system shall provide an admin interface for listing invitations.
- **REQ-11.4.2** [satisfied]: The system shall support sending invitations from the admin interface.
- **REQ-11.4.3** [satisfied]: The system shall support deleting invitations from the admin interface.

11.5 History and Monitoring

- **REQ-11.5.1** [satisfied]: The system shall provide a query history interface showing queries aggregated across selected repositories.

- **REQ-11.5.2** [satisfied]: The system shall display query history in a table format with sortable columns (timestamp, account, repository, signature, runtime).
- **REQ-11.5.3** [satisfied]: The system shall provide a graph view of query history plotting runtime (in seconds) against timestamp.
- **REQ-11.5.4** [satisfied]: The system shall make query history graph data-points clickable to display the query text with timestamp, account name, and repository name.
- **REQ-11.5.5** [satisfied]: The system shall provide a transaction history interface showing system-wide transaction history.
- **REQ-11.5.6** [satisfied]: The system shall display transaction history in a table format with sortable columns.
- **REQ-11.5.7** [satisfied]: The system shall provide an import history interface showing system-wide import history.
- **REQ-11.5.8** [satisfied]: The system shall display import history in a table format with pagination support.
- **REQ-11.5.9** [satisfied]: The system shall provide a graph view of import history plotting quad count against timestamp using a logarithmic scale.
- **REQ-11.5.10** [satisfied]: The system shall support tabbed sub-panes in history views (table view and graph view) with appropriate icons.

11.6 Admin Dashboard

- **REQ-11.6.1** [satisfied]: The system shall provide a dashboard interface with multiple panes for different administrative functions.
- **REQ-11.6.2** [satisfied]: The system shall support lazy-loading of admin pane content to reduce initial load time.

12. Data Visualization

12.1 Graph Rendering

- **REQ-12.1.1** [satisfied]: The system shall render SVG graphs for data visualization.
- **REQ-12.1.2** [satisfied]: The system shall automatically adjust horizontal scale based on earliest and latest timestamps in the data.
- **REQ-12.1.3** [satisfied]: The system shall automatically adjust vertical scale to maximum values, rounded to powers of ten.
- **REQ-12.1.4** [satisfied]: The system shall support logarithmic scaling for Y-axis when appropriate (e.g., import history).
- **REQ-12.1.5** [satisfied]: The system shall generate appropriate time grid markers (minutes, hours, days, or 6-month intervals) based on the overall time interval.
- **REQ-12.1.6** [satisfied]: The system shall place time grid markers and labels on the horizontal axis.
- **REQ-12.1.7** [satisfied]: The system shall provide adequate spacing between the horizontal axis and the image bottom boundary.

- **REQ-12.1.8** [satisfied]: The system shall use small dots (1px radius) and lines (1px width) for graph data points and paths.
- **REQ-12.1.9** [satisfied]: The system shall label graph axes appropriately (e.g., "Run Time (s)" for query history, "Quads" for import history, "Time" for horizontal axis).

Non-Functional Requirements

13. Performance

13.1 Response Time

- **REQ-13.1.1** [satisfied]: The system shall render initial page load within 2 seconds on standard broadband connections.
- **REQ-13.1.2** [satisfied]: The system shall update UI in response to user actions within 100ms for local operations.

13.2 Resource Usage

- **REQ-13.2.1** [satisfied]: The system shall minimize memory usage by cleaning up unused event listeners and DOM references.
- **REQ-13.2.2** [satisfied]: The system shall limit event log entries to prevent memory issues (default: 100 entries).

14. Browser Compatibility

14.1 Supported Browsers

- **REQ-14.1.1** [satisfied]: The system shall support modern browsers with ES6+ JavaScript support.
- **REQ-14.1.2** [satisfied]: The system shall support browsers with History API or fallback to hash-based routing.

15. Security

15.1 Authentication

- **REQ-15.1.1** [satisfied]: The system shall securely store authentication tokens (preferably in memory, with optional secure storage).
- **REQ-15.1.2** [satisfied]: The system shall not expose passwords in logs or error messages.
- **REQ-15.1.3** [satisfied]: The system shall include Bearer token authorization headers in API requests for authenticated operations.

15.2 Data Protection

- **REQ-15.2.1** [satisfied]: The system shall use HTTPS for all API communications.

- **REQ-15.2.2** [satisfied]: The system shall validate and sanitize user input before sending to the server.

16. Maintainability

16.1 Code Organization

- **REQ-16.1.1** [satisfied]: The system shall organize code into logical modules (routing, UI components, models, persistence).
- **REQ-16.1.2** [satisfied]: The system shall use consistent naming conventions and code style.

16.2 Documentation

- **REQ-16.2.1** [satisfied]: The system shall provide inline code documentation for complex logic.
- **REQ-16.2.2** [satisfied]: The system shall maintain API documentation for external integrations.

External Dependencies

17. External Services

17.1 Dydra API

- **REQ-17.1.1** [satisfied]: The system shall integrate with the Dydra service API for all data operations.
- **REQ-17.1.2** [satisfied]: The system shall handle API versioning and backward compatibility.

18. External Libraries

18.1 UI Libraries

- **REQ-18.1.1** [satisfied]: The system shall use Bootstrap CSS framework for styling.
- **REQ-18.1.2** [satisfied]: The system shall use jQuery UI for certain UI components.
- **REQ-18.1.3** [satisfied]: The system shall use YASQE (Yet Another SPARQL Query Editor) for SPARQL editing.
- **REQ-18.1.4** [satisfied]: 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

Summary of Changes from Original Requirements

New Requirements Added

- **Section 3.4.3-3.4.7**: Enhanced repository import capabilities (URL imports, file uploads, async imports, progress tracking, format auto-detection)
- **Section 3.5**: Repository export functionality
- **Section 3.6**: Repository operations (clear repository)
- **Section 3.7**: Repository collaboration management
- **Section 4.2.5-4.2.7**: Enhanced view management (open in pane/window)
- **Section 4.3.6-4.3.8**: Advanced query editor features (multiple tabs, drag-to-window, state preservation)
- **Section 5.2.6**: Cross-app navigation
- **Section 6.3.4**: Tabbed sub-panes
- **Section 11**: Complete administration interface section (new)
- **Section 12**: Data visualization section (new)
- **Section 15.1.3**: Bearer token authorization requirement

Requirements Status

- All original requirements from `requirements.md` have been retained
- All requirements are marked as **satisfied** based on current implementation
- No requirements have been marked as **deprecated** at this time
- No requirements are marked as **to be satisfied** as all identified features have been implemented