Pyxis: Open-Source Search Engine

System Documentation v1.0.0

Syed Abdullah Al Muyeed

Pyxis Labs

February 2026

Contents

1 Introduction

Pyxis is a privacy-focused search engine built on DuckDuckGo's API. It adds personal bookmarks with custom keyword search, collaborative group collections, and user-controlled data management. The entire platform is open-source under GPL-v3.

Unlike traditional search engines, Pyxis lets you tag bookmarks with keywords that appear first in search results. You can share bookmark collections with groups, track your search history, and delete everything whenever you want. There's no ad tracking or data selling.

2 User Experience

2.1 Public vs Account Access

You can use Pyxis without an account. Visit the site and search immediately across web, images, videos, and news. You get autocomplete suggestions and unlimited searches. Content filtering is applied universally. Nothing is saved when you search publicly.

Create an account to save searches, bookmark results with custom keywords, join groups, and get personalized autocomplete based on your history. Age verification from your birthdate controls content filtering levels. The system prompts you to sign up when you try to bookmark something.

2.2 Account Creation

Registration requires your name, birthdate, country, email, and password. Phone number is optional. Your email must be unique and serves as your login. Passwords are hashed before storage.

Users are categorized as either regular clients or admins. Admins can manage platform settings. Your account status can be active, inactive, or suspended. Birthdates determine content filtering: under 18 gets mandatory strict filtering, 18+ gets full control.

2.3 Search Features

Four search types are available: web (articles and websites), image (photos and graphics), video (video content), and news (current articles). All searches go through DuckDuckGo's API.

Autocomplete suggestions appear as you type, based on your search history if you're logged in. Every search is recorded with a timestamp in the searches table. Each clicked result gets saved separately in the histories table with the page title, URL, and icon.

You can delete individual searches or clear your entire history. The data is yours to manage.

2.4 Personal Keywords

Bookmark any website and assign custom keywords. When you search those keywords, your bookmarks appear at the top of results before any DuckDuckGo results.

If you bookmark your company wiki with the keyword "wiki", searching "wiki" shows it first. Frequently-used resources become instantly accessible through your own vocabulary. This transforms generic search into personalized navigation.

Each bookmark stores a title, URL, icon, keywords, and sort order. Bookmarks can be personal or assigned to groups for collaboration.

2.5 Collaborative Groups

Groups are shared bookmark collections. Create a group, add bookmarks, and invite members. Everyone in the group contributes to and accesses the shared collection. Each group has a unique share code for easy joining.

Group creators automatically become admins. Admins can manage members, moderate bookmarks, and control group settings. Regular members can add bookmarks and view the collection.

Member status tracks whether someone is pending (invited but not joined), active (participating), or removed (kicked). Groups are useful for research teams, study groups, or any collaborative knowledge gathering.

2.6 Notifications

Two notification types exist: user-to-user (group invitations) and system-generated (account alerts, security notices). All notifications show who sent them, include action links when relevant, and track read status.

Group invitations show the sender, group details, and buttons to accept or decline. System notifications alert you to account events or platform updates. Notifications are stored with timestamps and can be marked as read.

2.7 User Settings

Preferences are stored as JSON, allowing flexible customization without database changes. Settings include content filtering levels, search preferences (default type, results per page, safe search), privacy controls, and notification preferences.

Under-18 users have locked content filtering they can't disable. Adults control all filtering settings. The JSON structure means new preference types can be added without modifying the database schema.

2.8 Activity and Security Logging

The logs table tracks three types of events: access (login/logout with IP and device info), group (membership changes, bookmark additions), and system (platform events).

Access logs include IP addresses, browser information, operating system, and device type, all stored in a JSON details field. This creates an audit trail to spot unauthorized access from unfamiliar locations or devices.

Group logs track who did what when: adding bookmarks, inviting members, or moderating content. Everything is timestamped and attributed to specific users for accountability.

3 Database Architecture

3.1 Design Principles

The database uses 10 normalized tables with consistent naming conventions. Every table has uuid (primary key), id (auto-increment), created_at, and updated_at fields. Foreign keys enforce referential integrity with cascade deletes.

UUIDs provide globally unique identifiers across distributed systems. Integer IDs offer simple sequential references. Both are unique within each table.

Indexes optimize common queries: user searches, group member lookups, notification retrieval, and access log analysis. JSON fields in logs and settings provide schema flexibility.

3.2 Table Schemas

3.2.1 users

Stores all account information. Each user has a name, birthdate, country, optional phone, unique email, hashed password, role (admin or client), and status (active, inactive, or suspended).

Column	Type	Description
uuid	VARCHAR(255)	Primary key, unique identifier
id	INT	Auto-increment, unique
name	VARCHAR(255)	User's full name
dob	DATE	Birthdate for age verification
country	VARCHAR(100)	Country of residence
phone	VARCHAR(20)	Optional phone number
email	VARCHAR(255)	Unique email for login
password	VARCHAR(255)	Hashed password
role	ENUM	'admin' or 'client'
status	ENUM	'active', 'inactive', or 'suspended'
created_at	DATETIME	Account creation timestamp
updated_at	DATETIME	Last modification timestamp

Email and phone number uniqueness is enforced at the database level. Passwords are never stored in plain text. The role determines platform permissions, and status controls account access.

3.2.2 searches

Records every search query. Each search links to the user who performed it and includes the search text and timestamp.

Column	Type	Description
uuid	VARCHAR(255)	Primary key
id	INT	Auto-increment, unique
user_uuid	VARCHAR(255)	Foreign key to users.uuid
query	VARCHAR(255)	The search text
created_at	DATETIME	Search timestamp
updated_at	DATETIME	Last modification

Searches are linked to users via user_uuid. When a user is deleted, all their searches are automatically removed (cascade delete).

3.2.3 histories

Tracks clicked search results. When you click a result from a search, it gets saved here with the search reference, page title, URL, and icon.

Column	Type	Description
uuid	VARCHAR(255)	Primary key
id	INT	Auto-increment, unique
user_uuid	VARCHAR(255)	Foreign key to users.uuid
search_uuid	VARCHAR(255)	Foreign key to searches.uuid
title	VARCHAR(255)	Page title
url	LONGTEXT	Full URL
icon_url	LONGTEXT	Site icon/favicon
created_at	DATETIME	Click timestamp
updated_at	DATETIME	Last modification

This creates a browsing trail connected to specific searches. You can see what you clicked from which query. Both user and search deletions cascade to remove related histories.

3.2.4 bookmarks

Stores user bookmarks with personalized keywords. Bookmarks can be personal (group_uuid is NULL) or shared within groups.

Column	Type	Description
uuid	VARCHAR(255)	Primary key
id	INT	Auto-increment, unique
user_uuid	VARCHAR(255)	Foreign key to users.uuid
group_uuid	VARCHAR(255)	Optional foreign key to groups.uuid
title	VARCHAR(255)	Bookmark display name
url	LONGTEXT	Website URL
icon_url	LONGTEXT	Site icon for visual recognition
keywords	VARCHAR(255)	Custom search keywords
sort_order	INT	Display position (must be unique)
created_at	DATETIME	Creation timestamp
updated_at	DATETIME	Last modification

Keywords enable personalized search prioritization. When you search a keyword, matching bookmarks appear first. Sort order must be unique to prevent conflicts in bookmark lists. Group bookmarks belong to the group but are created by individual users.

3.2.5 groups

Collaborative bookmark collections. Each group has a creator, name, icon, and unique share code.

Column	Type	Description
uuid	VARCHAR(255)	Primary key
id	INT	Auto-increment, unique
creator_uuid	VARCHAR(255)	Foreign key to users.uuid
name	VARCHAR(255)	Group name
icon	VARCHAR(255)	Group icon URL
share_code	VARCHAR(50)	Unique code for joining
created_at	DATETIME	Creation timestamp
updated_at	DATETIME	Last modification

The creator automatically becomes a group admin. Share codes must be unique across all groups. When a creator's account is deleted, the entire group is removed.

3.2.6 members

Tracks group membership with roles and status. Each member record connects a user to a group.

Column	Type	Description
uuid	VARCHAR(255)	Primary key
id	INT	Auto-increment, unique
user_uuid	VARCHAR(255)	Foreign key to users.uuid
group_uuid	VARCHAR(255)	Foreign key to groups.uuid
role	ENUM	'admin' or 'member'
status	ENUM	'pending', 'active', or 'removed'
created_at	DATETIME	Membership creation
updated_at	DATETIME	Last modification

A user can only belong to a group once (enforced by unique constraint on user_uuid + group_uuid combination). Status 'pending' means invited but not accepted. Status 'removed' means kicked from the group. Admins have elevated permissions for group management.

3.2.7 invites

Records group invitations. Each invite tracks who sent it, who received it, and the current status.

Column	Type	Description
uuid	VARCHAR(255)	Primary key
id	INT	Auto-increment, unique
sender_uuid	VARCHAR(255)	Foreign key to users.uuid
receiver_uuid	VARCHAR(255)	Foreign key to users.uuid
type	ENUM	Currently only 'group'
status	ENUM	'pending', 'accepted', or 'declined'
created_at	DATETIME	Invitation sent
updated_at	DATETIME	Last modification

The type field allows for future invitation types beyond groups. Status tracks the invitation lifecycle. Declined invitations remain for audit purposes rather than being deleted.

3.2.8 notifications

User notification inbox. Handles both user-to-user notifications (like group invites) and system-generated alerts.

Column	Type	Description
uuid	VARCHAR(255)	Primary key
id	INT	Auto-increment, unique
sender_uuid	VARCHAR(255)	Optional foreign key to users.uuid
receiver_uuid	VARCHAR(255)	Foreign key to users.uuid
title	VARCHAR(255)	Notification heading
message	VARCHAR(255)	Notification content
action_url	LONGTEXT	Optional link for action
is_read	TINYINT(1)	0 = unread, 1 = read
type	ENUM	'system', 'invite', or 'group'
created_at	DATETIME	Notification created
updated_at	DATETIME	Last modification

Sender can be NULL for system notifications. Type categorizes notifications for filtering and display. Action URLs provide direct links to relevant pages (accept invite, view group, etc.).

3.2.9 logs

Unified logging for access events, group activities, and system operations. Uses JSON for flexible detail storage.

Column	Type	Description
uuid	VARCHAR(255)	Primary key
id	INT	Auto-increment, unique
user_uuid	VARCHAR(255)	Optional foreign key to users.uuid
type	ENUM	'access', 'group', or 'system'
details	JSON	Event-specific information
created_at	DATETIME	Event timestamp
updated_at	DATETIME	Last modification

The details field stores event-specific data in JSON format. For access logs, this includes IP address, user agent string, browser type, operating system, and device type. For group logs, it stores actions like member additions or bookmark changes. For system logs, it records platform events.

Example access log details:

```
{
  "action": "login",
  "ip_address": "192.168.1.1",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)...",
  "browser": "Chrome",
  "os": "Windows 10",
  "device_type": "desktop"
}
```

Example group log details:

```
{
  "action": "member_added",
  "group_uuid": "abc123",
  "new_member_uuid": "def456",
  "added_by": "ghi789"
}
```

This approach provides schema flexibility. New fields can be added to JSON without database migrations.

3.2.10 settings

User preference storage using JSON. One settings record per user.

Column	Type	Description
uuid	VARCHAR(255)	Primary key
id	INT	Auto-increment, unique
user_uuid	VARCHAR(255)	Foreign key to users.uuid (unique)
preferences	JSON	All user preferences
created_at	DATETIME	Settings created
updated_at	DATETIME	Last modification

User_uuid is unique, enforcing one settings record per user. The preferences JSON stores all customization options.

Example preferences structure:

```
{
  "content_filter": "moderate",
  "search_preferences": {
    "default_type": "web",
    "safe_search": true,
    "results_per_page": 20
  },
  "notifications": {
    "email": true,
    "push": false
  },
  "privacy": {
    "show_search_history": true,
    "data_retention_days": 90
  }
}
```

JSON validation ensures proper structure. New preference types can be added without schema changes.

3.3 Relationships and Integrity

All foreign keys cascade on delete. Deleting a user removes all their searches, bookmarks, group memberships, notifications, logs, and settings. Deleting a group removes all its members, bookmarks, and related logs.

Unique constraints prevent duplicate emails, share codes, bookmark sort orders, and group memberships. NOT NULL constraints ensure critical data is always present. ENUM fields restrict values to predefined options.

Every table includes created_at (set once) and updated_at (auto-updates on modification) timestamps for complete audit trails.

3.4 Indexing

Primary keys (uuid) and unique fields are automatically indexed. Foreign keys have indexes for efficient joins. Common query patterns are optimized: finding user searches, loading group members, fetching unread notifications, and analyzing access logs by timestamp.

The database handles millions of records efficiently through proper indexing and normalization.

4 Security and Privacy

Passwords are hashed before storage using secure algorithms. Database administrators cannot see user passwords. IP addresses and device information in access logs help identify suspicious activity from unfamiliar locations.

Data collection is minimal and purpose-driven. Birthdates verify age for content filtering. Search histories power autocomplete suggestions, not advertising. IP addresses support security monitoring with limited retention.

Users control their data completely. Delete individual searches, clear entire histories, or remove accounts with all associated data. Nothing is sold to third parties. No targeted advertising exists.

Age-based content filtering is automatic. Under-18 users get mandatory strict filtering across all search types. Adult users (18+) have full control to customize or disable filtering entirely.

5 Technical Considerations

The normalized structure minimizes data redundancy while maintaining query performance. JSON fields in logs and settings provide extensibility without schema changes. Cascade deletes maintain referential integrity automatically.

The architecture supports horizontal scaling through potential sharding strategies. Clean table separation enables distributed systems. Indexed foreign keys optimize distributed queries.

Future features can be added without major rewrites. New preference types go into the settings JSON. New log types use the existing details field. New invitation types extend the invites table. The modular design accommodates growth.

6 Open Source License

Pyxis is licensed under GNU General Public License v3.0. You can use, study, modify, and distribute the code freely. All derivative works must also be open-source under GPL-v3. The source code and this documentation are publicly available.

Contributions are welcome: bug reports, feature requests, code improvements, documentation updates, and translations. All contributions must maintain GPL-v3 licensing.