# Software Requirements Spec. (SRS): New features for myHAW

## 0 Document Information

This document is based on IEEE Standard 830.

### 0.1 List of Authors

| Group/Team | Names |
|---|---|
| Group-03 | Thilakraj Soundararajan |
| <Your team name> | Shifat jahan shama |

### 0.2 Revision History

| Version | Date | Author | Notes |
|---|---|---|---|
| 0.1 | | Shifat jahan shama | Wrote down the purpose and all the functional requirements. |
| 0.2 | | Thilakraj Soundararajan | Added Non-functional requirements and made some changes to functional requirements because one of the requirements was actually non-functional. |
| 0.3 | | Thilakraj Soundararajan | Changed and added some functional requirements related to filter, autocorrection and admin indexing. |
| | | Shifat jahan shama | Added nonfunctional requirements related to crawler and Security |
| 0.4 | | Thilakraj Soundararajan | Created the sequence diagrams for 3 cases and also the UML diagram for the entire system based on sequence diagrams and domain model |
| 0.5 | | Thilakraj Soundararajan | Changed the UML Diagram with clear methods and relations to be more consistent with SD. |
| | | | |
| | | | |
| | | | |

## Content

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to define the software requirements for a web-based Internet search engine. The system aims to deliver fast, relevant, and user-friendly search capabilities by modeling interactions between two primary actors: the **end user**, who performs search queries, and the **website owner**, who submits and manages content for indexing. This specification outlines functional behavior, performance expectations, and system constraints, serving as a foundational reference for development, testing, and future enhancements.

## 1.2 Scope

The system is a web-based search engine designed to enable users to retrieve relevant information from a continuously updated index of web content. Users can submit queries, receive ranked results, and apply filters to refine those results based on criteria such as date, relevance, or content type. Website owners can submit content for indexing and monitor how their pages appear in search results. Core functionalities include web crawling, indexing, query processing, result ranking, and result filtering. The system is designed for high performance, scalability, and ease of use.

## 1.3 Glossary

The following abbreviations and domain-specific terms are used throughout the document.

| Abbreviation | Term | Description | Reference |
|---|---|---|---|

| | | | |
|---|---|---|---|
| UC | Use Case | A set of interactions between actors and the system that achieve a specific goal. | UML Standards |
| NFR | Non-Functional Requirement | A system constraint defining performance, reliability, or usability rather than specific behaviors. | IEEE SRS Guidelines |
| SRS | Software Requirements Specifications | A document that outlines the functional and non-functional requirements of a software system. | IEEE SRS Standard (830-1998) |
| Actor | Actor | A user or external entity that interacts with the system to achieve a specific objective. | UML Use Case Modeling |
| SERP | Search Engine Results Page | The interface displaying ranked search results in response to a user's query. | Search Engine Design Docs |
| Crawling | Crawling | The automated process of systematically browsing and collecting data from web pages for indexing. | Information Retrieval Systems |
| Indexing | Indexing | The automated process of organizing and storing web data for efficient retrieval. | Information Retrieval Systems |

# 2 General description

This section of the SRS describes the general requirements that drive the design of the software system. The goal is not to state specific requirements, but rather to provide context to make those requirements easier to understand.

## 2.1 Product perspective

This product is a self-contained web-based Internet search engine designed to allow users to query, retrieve, and interact with web content. It is not a component of a larger system but functions independently with a modular architecture that supports future extensibility (e.g. additional search features, analytics tools). The product includes distinct user-facing and admin-facing interfaces. Users interact with the system to perform content searches and refine results, while administrators manage content indexing and system updates.

Major subsystems include:

- **Search Interface Module** – accepts and validates queries

- **Indexing & Crawling Subsystem** – stores and updates searchable content
- **Results Engine** – processes queries, ranks and displays results
- **Filtering & Interaction Layer** – supports opening links, filtering outputs, and UI feedback
- **Admin Control Panel** – for index submission, updates and backend operations

## 2.2 Product functions

The system supports the following core functions:

### UC-1: Submit Search Query

**Actors**: User
**Description**: The user submits a search query and receives ranked results based on the keywords in the query.
**Triggered by**: FR-1 (Search Query Submission)
**Steps**:

→ User types a query in the search box.
→ The system processes the query and returns relevant results.
→ If no results are found, the system displays "No Results Found".

### UC-2: Correct Typos and Autocorrect Query

**Actors**: User
**Description**: The system trims unnecessary typos and autocorrects spelling errors in the query to process the keywords properly.
**Triggered by**: FR-2 (Query Correction) **Steps**:

→ User submits a query with possible typos.
→ The system corrects the spelling errors and processes the query correctly.

### UC-3: Display No Results Found

**Actors**: System
**Description**: If the query returns no results, the system displays "No Results Found."
**Triggered by**: FR-3 (No Results Found Message)
**Steps**:

→ User submits a query.
→ The system checks the index and finds no matching results.
→ The system displays the message "No Results Found."

### UC-4: Display Search Results

**Actors**: User
**Description**: The system displays search results with titles, snippets, and navigation controls.
**Triggered by**: FR-4 (Search Results Display)
**Steps**:

→ User submits a query.
→ The system displays results with titles, snippets, and navigation controls.
→ The user can navigate through the results.

### UC-5: Filter and Sort Search Results

**Actors**: User
**Description**: The system allows users to filter results based on criteria such as language and sort by relevance or last updated time.
**Triggered by**: FR-5 (Filter and Sort Results)
**Steps**:

→ User submits a query.
→ The system displays results.
→ The user applies filters and sorts the results.
→ The system re-displays results based on the selected criteria.

### UC-6: Open Search Result in New Tab

**Actors**: User
**Description**: The system allows users to open selected results in a new browser tab or window.
**Triggered by**: FR-6 (Open Search Result in New Tab)
**Steps**:

→ User clicks on a search result.
→ The system opens the result in a new tab or window.

### UC-7: Open Multiple Tabs

**Actors**: User
**Description**: The system allows users to open multiple search results in different tabs.
**Triggered by**: FR-7 (Open Multiple Tabs)
**Steps**:

→ User clicks on multiple search results.
→ The system opens each result in a new tab.

### UC-8: Display Error Message for No Results

**Actors**: System
**Description**: If no results are found or if there are errors with the query, the system displays an error

message.
**Triggered by**: FR-8 (Display Error Message)
**Steps**:

→ User submits a query.
→ The system processes the query.
→ If results are not found or an error occurs, the system displays the appropriate error message ("Result not found").

### UC-9: Admin Submits Website for Crawling

**Actors**: Admin (Website Owner)
**Description**: The website owner submits their website for crawling via the Admin Control Panel.
**Triggered** by: FR-11 (Submit Website for Indexing)
Steps:

→ Admin logs into the Admin Control Panel.
→ Admin submits the website URL for crawling and indexing.
→ The system initiates the crawling process.

### UC-10: Admin Submits Sitemap for Indexing

**Actors**: Admin (Website Owner)
**Description**: The website owner submits a sitemap containing all URLs to be indexed.
**Triggered by**: FR-12 (Submit Sitemap for Indexing)
**Steps**:

→ Admin logs into the Admin Control Panel.
→ Admin submits the website sitemap containing all URLs.
→ The system processes the sitemap and initiates the crawling and indexing.

### UC-11: Website Support for Crawling

**Actors**: Admin (Website Owner)
**Description**: The website must be configured to allow crawling by the search engine’s crawler.
**Triggered by**: FR-13 (Website Must Support Crawling)
**Steps**:

→ The website owner configures the website to allow access for the crawler.
→ The crawler scans and indexes the website’s content.

### UC-12: Search Engine Performs Crawling

**Actors**: System (Search Engine)
**Description**: The search engine must perform the crawling operation to index the website's content.

**Triggered by**: FR-14 (Search Engine Performs Crawling)
**Steps**:

→ The system performs the crawling operation on the submitted website.
→ The crawler indexes the website's content.

**UC-13: Periodic Crawling**

**Actors**: System (Search Engine)
**Description**: The system must automatically initiate periodic crawling based on the selected frequency (6 months, 12 months).
**Triggered by**: FR-3 (Periodic Crawling)
**Steps**:

→ The system checks the periodic crawling schedule (6 months or 12 months).
→ The crawler performs the crawling operation for the website according to the schedule.
→ The system indexes the newly crawled content.

Each of these functions corresponds to a clearly defined use case and is traceable to specific functional requirements in the system.

## 2.3 User characteristics

The system is designed to support two primary user classes:

- **End Users (General Users):** Individuals using the search engine to find and access information. They are expected to have basic web browsing experience but do not require technical expertise. Their interactions are limited to submitting queries, browsing results, applying filters, and opening links. Usability, responsiveness, and clear feedback are essential for this group.

- **Administrators (Website Owners):** Users responsible for managing content visibility through index updates. This group may include content providers or technical staff with elevated permissions. They are assumed to have familiarity with content management workflows and system settings. Security, authorization, and operational transparency are critical for their tasks.

- **Search Engine Authority:** This group provides the crawler, which automatically browses and indexes content from websites. They also provide an admin panel that allows website owners (administrators) to submit their websites for indexing. This helps ensure that the relevant content is visible and accessible to end users. The crawler performs regular updates to keep the search engine results up to date. This authority is also responsible for maintaining the search engine functionality that end users interact with to perform their queries.

## 2.4 Constraints

The following constraints apply to the development and operation of the system:

- **Real-time Performance:**
  The system must return query results within an acceptable response time (e.g., under 1 second for standard queries).

- **Indexing Limitations:**
  Index updates should be handled during off-peak hours or incrementally to avoid service degradation during peak user activity.

- **Security & Authorization:**
  Only authenticated admin users are allowed to update the index or access administrative controls.

- **Scalability:**
  The system should support a scalable architecture that allows horizontal expansion as the indexed content volume grows.

- **Technology Stack:**
  The system should be implemented using open-source, web-compatible technologies to ensure portability and ease of deployment.

## 2.5 Assumptions and dependencies

- It is assumed that users will access the system through modern web browsers that support JavaScript and responsive design.
- The system depends on stable internet connectivity for all user interactions and for fetching up-to-date indexed content.
- The content submitted for indexing is assumed to be safe, relevant, and correctly formatted by the administrator or content provider.
- External APIs or services used for language detection, content parsing, or performance tracking are assumed to be available and stable.
- Security and authentication infrastructure (e.g., HTTPS, session management) are assumed to be properly configured and maintained.

# 3 External interface requirements

**User Interface (UI):**

- The system shall provide a web-based graphical interface accessible via standard browsers.
- The interface includes input fields for search queries e.g. search bar where the query is submitted, field showing controls for filtering results, and a result display section with pagination.

**Hardware Interfaces:**

- No specialized hardware interfaces are required. The system is hosted on standard cloud or on-premise web servers.

**Software Interfaces:**

- The system may interface with external modules for tasks such as:
    - Language detection libraries
    - Content parsing tools
    - Admin authentication systems

**Communication Interfaces:**

- Communication with users occurs over HTTPS.
- Admin panel access requires secure login protocols and role-based permissions.
- All data transfers between system modules and services should use REST APIs or equivalent secure communication protocols.

## 4 Functional requirements

<Describe the functional requirements and their priorities. Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

| Identifier | Priority (MoSCoW) | Requirement |
|---|---|---|
| **REQ-1** | MUST | The system must allow users to submit a search query and receive ranked results based on the keywords in the search query from the user. |
| **REQ-2** | SHOULD | The system should trim the unnecessary typo and autocorrect spellings in the query allowing the system to process the keywords properly. |
| **REQ-3** | MUST | The system must display “No Results Found” if no results are found for a query or when the query is empty. |

| | | |
|---|---|---|
| **REQ-4** | MUST | The system must display search results with titles, snippets, and navigation controls. |
| **REQ-5** | SHOULD | The system should allow users to filter results based on criteria such as language and sort search results by at least relevance and last updated time. |
| **REQ-6** | MUST | The system must allow users to open a selected result in a new browser tab or window. |
| **REQ-7** | SHOULD | The system should allow the user to open multiple tabs at the same time. |
| **REQ-8** | SHOULD | The system should display error message “Result not found” for unavailable pages or failed updates after applying filter. |
| **REQ-9** | SHOULD | The system should display query suggestions or autocomplete options after the user has typed at least 3 characters in the search box. |
| **REQ-10** | MUST | The system must limit the number of search results shown on one page to 20 and provide controls to view additional pages or load more results. |
| **REQ-11** | MUST | The system shall provide an Admin Control Panel where the website owner can submit a website for indexing. |
| **REQ-12** | SHOULD | The system shall allow the website owner to submit a sitemap containing all URLs to be indexed. |
| **REQ-13** | MUST | The website must be designed to handle crawling. The website should allow the crawler to access and index all public pages and content. |
| **REQ-14** | MUST | The search engine must provide the ability to perform website crawling. |
| **REQ-15** | MUST | The crawler must be triggered when the website is first submitted for indexing, and again after each update. |
| **REQ-16** | MUST | Indexing must occur promptly after crawling, ensuring that content is available in search results quickly. |
| **REQ-17** | MUST | The system shall initiate indexing promptly after crawling to ensure content is available in search results. |
| **REQ-18** | SHOULD | The Admin Control Panel should provide website owners with two or three pre-configured periodic crawling frequency options for their website. These options shall be 6 months, 12 months |

## 5 Non-functional requirements

| Identifier | Requirement | Measurable |
|---|---|---|
| NFR-1 | Performance - The system must process and return search results within 2 seconds of submitting a query. | Response time must be tracked and analysed under typical load conditions to ensure the 2-second threshold is met 95% of the time. |
| NFR-2 | Scalability - The system should be able to handle up to 1000 concurrent users performing search queries. | Load testing should be conducted to ensure the system can maintain 95% uptime under the peak load of 1000 concurrent users. |
| NFR-3 | Usability - The system must provide a 90% user satisfaction rate in terms of ease of use for submitting queries and navigating results. | Conduct user surveys or usability testing with at least 100 users, aiming for at least 90% positive feedback regarding usability. |
| NFR-4 | Availability - The system must maintain 99.9% uptime over a 30-day period. | Regular system monitoring should be implemented to track uptime and downtime, ensuring the system adheres to the 99.9% availability requirement. |
| NFR-5 | Security – The must check if the data is encrypted to make sure that the data from the user is protected | The system shall run an automated check once every 24 hours to verify that all configured data fields marked as "sensitive" are stored in encrypted form and log the result of this check. |
| NFR-6 | Trigger - The crawler must be triggered within a given timeframe once a website has been submitted for indexing | Trigger time within 30 seconds of submission. Update detection within 10 minutes of content changes. |

| | | |
|---|---|---|
| NFR-7 | Operation - The crawler must operate without significantly impacting website performance for end users. | No increase in page load time by more than 5%, crawler uses <20% CPU and <10% memory, 1000 concurrent users. |
| NFR-8 | Efficiency - The crawler should be efficient in terms of resource usage, supporting both full and incremental crawls. | Incremental crawl resource usage 80% less than full crawl, 100 pages per minute full crawl speed, <2% duplicate content. |
| NFR-9 | Indexing Occurrence - Indexing must occur right after crawling has been done, ensuring that content is available in search results quickly. | Indexing completed within 1 hour of crawl, search results updated within 30 minutes of indexing. |
| NFR-10 | Traffic Handling - The system must handle high traffic and scaling requirements as needed. | Handle up to 5000 pages concurrently, 50 simultaneous crawls, load balancing with <75% CPU utilization per server. |

# 6 Traceability Matrix

| Requirements | UC-01 | UC-02 | UC-03 | UC-04 | UC-05 | UC-06 | UC-07 | UC-08 | UC-09 | UC-10 | UC-11 | UC-12 | UC-13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REQ-01 | x | | | | | | | | | | | | |
| REQ-02 | | x | | | | | | | | | | | |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REQ-03 | | | x | | | | | | | | | | |
| REQ-04 | | | | x | x | | | | | | | | |
| REQ-05 | | | | | x | | | | | | | | |
| REQ-06 | | | | | | x | | | | | | | |
| REQ-07 | | | | | | | x | | | | | | |
| REQ-08 | | | | | | | | x | | | | | |
| REQ-09 | | | | | | | | | x | | | | |
| REQ-10 | | | | | | | | | | x | | | |
| REQ-11 | | | | | | | | | | | x | | |
| REQ-12 | | | | | | | | | | | | x | |
| REQ-13 | | | | | | | | | | | | | x |
| REQ-14 | | | | | | | | | | | | x | |
| REQ-15 | | | | | | | | | x | | | | |
| REQ-16 | | | | | | | | | | | | x | x |
| REQ-17 | | | | | | | | | | | | x | x |
| REQ-18 | | | | | | | | | | | | | x |

# 7 Domain Model

In this section, the conceptual structure of the *Search Engine System* is described.
The domain model has been derived from the use cases (UC-1 … UC-13) and the functional requirements of the SRS. The model is intended to capture the main concepts of the system, their responsibilities and their relationships, without committing to a specific technical implementation.

Three external actors are involved in the domain:

- **WebUser**, who performs searches and views results.
- **Owner**, who registers websites and submits sitemaps.
- **SEA & Crawler**, which provides and operates the crawling infrastructure.

Within the system boundary, one boundary object, one controller object and several entities are identified.

## Domain Concepts and Stereotypes

The domain concepts are grouped according to the Boundary–Control–Entity pattern.

**Boundary :**

- **SearchInterface** – user-facing interface through which search requests are entered and results are displayed.

**Control :**

- **SearchController** – controller responsible for coordinating the processing of search requests.

**Entities :**

- **Query** – representation of a concrete search request issued by a WebUser.
- **ResultList** – collection of results returned for a single Query.
- **ResultItem** – individual entry within a ResultList.
- **SearchIndex** – logical index containing information about indexed pages.

- **WebPage** – single web page that can be crawled and indexed.
- **CrawlJob** – execution of the crawler over a website or a set of pages.
- **Website** – website that is registered by an Owner for crawling.
- **Sitemap** – sitemap file listing URLs that belong to a Website.

## Responsibilities of Domain Concepts

The main responsibilities of each concept are summarised below. The descriptions are kept implementation-independent and formulated in terms of what information is held or what role is fulfilled in the domain.

| **Concept** | **Responsibilities** |
|---|---|
| **SearchInterface** (boundary) | The SearchInterface is used for the entry of query text by the WebUser and for the presentation of search results, including messages in the case of empty result sets. |
| **SearchController** (control) | Search requests are received by the SearchController from the SearchInterface. For each request, a corresponding Query is created, the SearchIndex is consulted, and a ResultList composed of ResultItems is produced for presentation. |
| **Query** (entity) | A Query stores the text of a search request together with its language and timestamp and serves as the logical link between the WebUser's request and the produced ResultList. |
| **ResultList** (entity) | A ResultList represents the complete response to one Query. It records the total number of hits and the criteria by which the list is filtered or sorted. |
| **ResultItem** (entity) | Each ResultItem represents a single visible hit in the search results. It contains the title, snippet, URL and ranking score of the matched WebPage. |
| **SearchIndex** (entity) | The SearchIndex maintains the searchable collection of indexed WebPages. It is consulted whenever a Query is processed and is updated whenever new or changed WebPages are discovered by CrawlJobs. |
| **WebPage** (entity) | A WebPage describes a specific page that can be crawled and indexed. It records URL, title, content and current index status. |

| Concept | Responsibilities |
|---|---|
| **CrawlJob** (entity) | A CrawlJob represents one execution of crawling against a website or a group of pages. It records start and end time, the number of pages crawled and whether the job is periodic. |
| **Website** (entity) | A Website represents a site registered by an Owner. It groups WebPages and CrawlJobs and records domain, status and last crawl time. |
| **Sitemap** (entity) | A Sitemap represents a sitemap file submitted by an Owner. It provides structured URL lists that can be used as input when CrawlJobs are defined. |

## Relationships Between Concepts

The most relevant relationships between the concepts are described below.
Two relationships are explicitly marked with the stereotypes <<include>> and <<extend>>
to emphasise the integration between search and crawling.

| From → To | Description |
|---|---|
| **WebUser → SearchInterface** | The SearchInterface is used by the WebUser to submit queries and to inspect the returned ResultList. |
| **SearchInterface → SearchController** | Search requests and user options are forwarded by the SearchInterface to the SearchController. |
| **SearchController → Query** | For each incoming search request, a Query instance is created and filled with the corresponding data. |
| **Query → ResultList** | Each Query is associated with exactly one ResultList that captures the response to that query. |
| **ResultList → ResultItem** | A ResultList is composed of multiple ResultItems that are displayed to the WebUser. |
| **SearchController → SearchIndex** (<<include>>) | Whenever a Query is processed, the SearchIndex is **included** in the processing pipeline and is consulted to obtain matching pages. |

| **From → To** | **Description** |
|---|---|
| **SearchIndex → WebPage** | The SearchIndex maintains references to all WebPages that have been indexed. |
| **ResultItem → WebPage** | Each ResultItem refers to the WebPage that satisfied the Query. |
| **Owner → Website** | One or more Websites are registered by an Owner for crawling by the search engine. |
| **Owner → Sitemap** | Sitemaps are submitted by the Owner and are associated with their corresponding Website. |
| **Website → WebPage** | A Website contains many WebPages; each WebPage belongs to exactly one Website. |
| **Website → CrawlJob** | CrawlJobs are scheduled for a Website (e.g. initial crawl or periodic recrawl). |
| **Sitemap → CrawlJob** | URLs listed in a Sitemap are used when CrawlJobs are configured for that Website. |
| **CrawlJob → WebPage** | During a CrawlJob, multiple WebPages are discovered and crawled. |
| **CrawlJob → SearchIndex** (<<extend>>) | After completion of a CrawlJob, the SearchIndex is **extended** or updated with the newly discovered or modified WebPages. |
| **SEA & Crawler → CrawlJob** | Execution of CrawlJobs is performed under the responsibility of the external actor SEA & Crawler. |

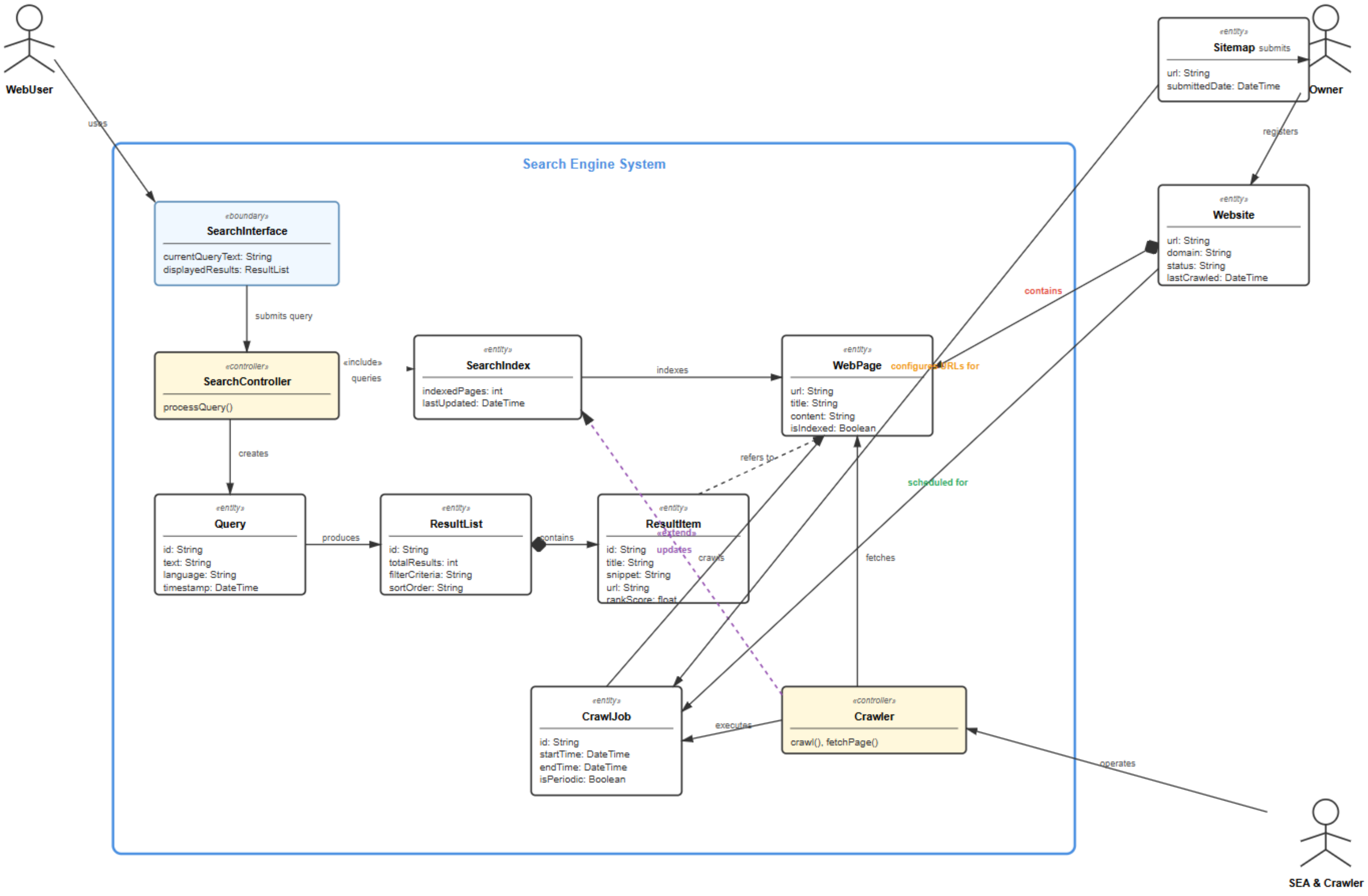
## Attributes of Entities

For each entity, only the principal attributes that are relevant at domain level are listed.
Exact data types and storage mechanisms are left open.

| **Entity** | **Key attributes** |
|---|---|
| **Query** | id, text, language, timestamp |
| **ResultList** | id, totalResults, filterCriteria, sortOrder |
| **ResultItem** | id, title, snippet, url, rankScore |

| Entity | Key attributes |
|---|---|
| SearchIndex | indexedPages (number or collection reference), lastUpdated |
| WebPage | url, title, content, isIndexed |
| CrawlJob | id, startTime, endTime, pagesCrawled, isPeriodic |
| Website | url, domain, status, lastCrawled |
| Sitemap | url, submittedDate |

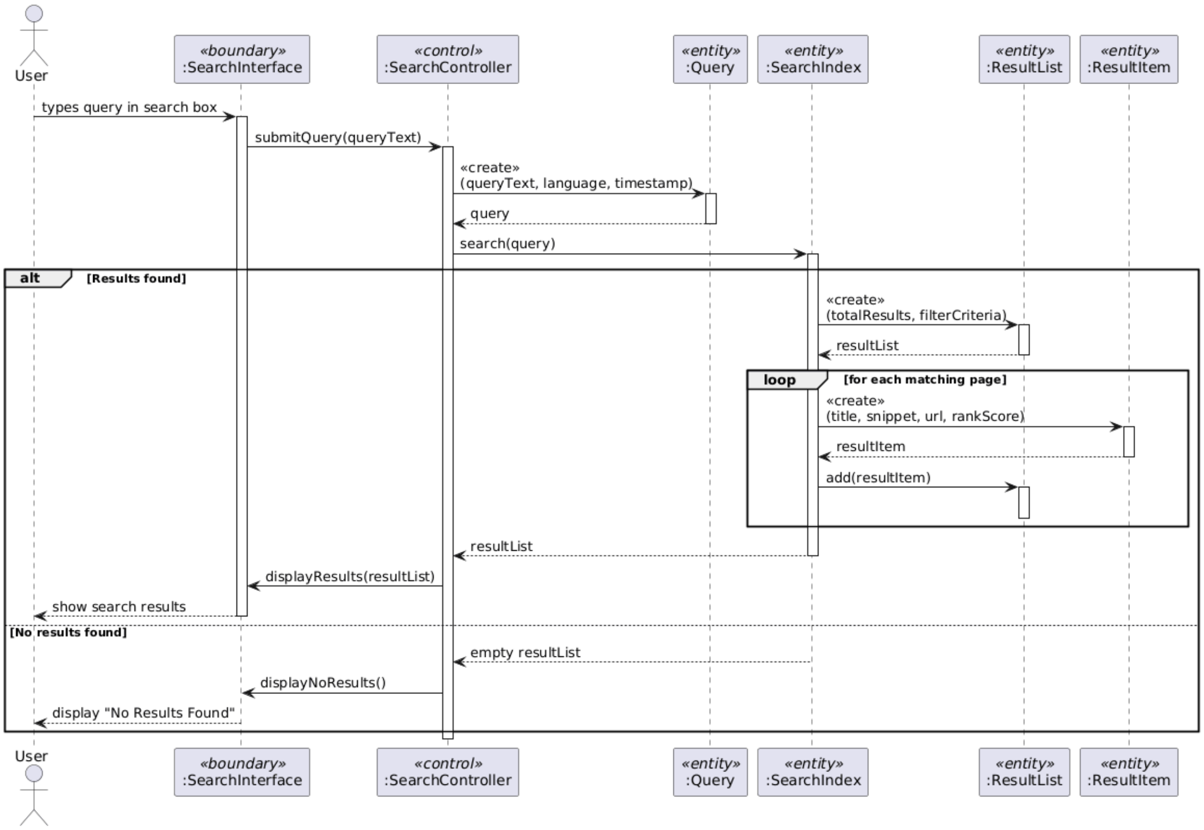
## Domain Model Diagram



## SEQUENCE DIAGRAMS

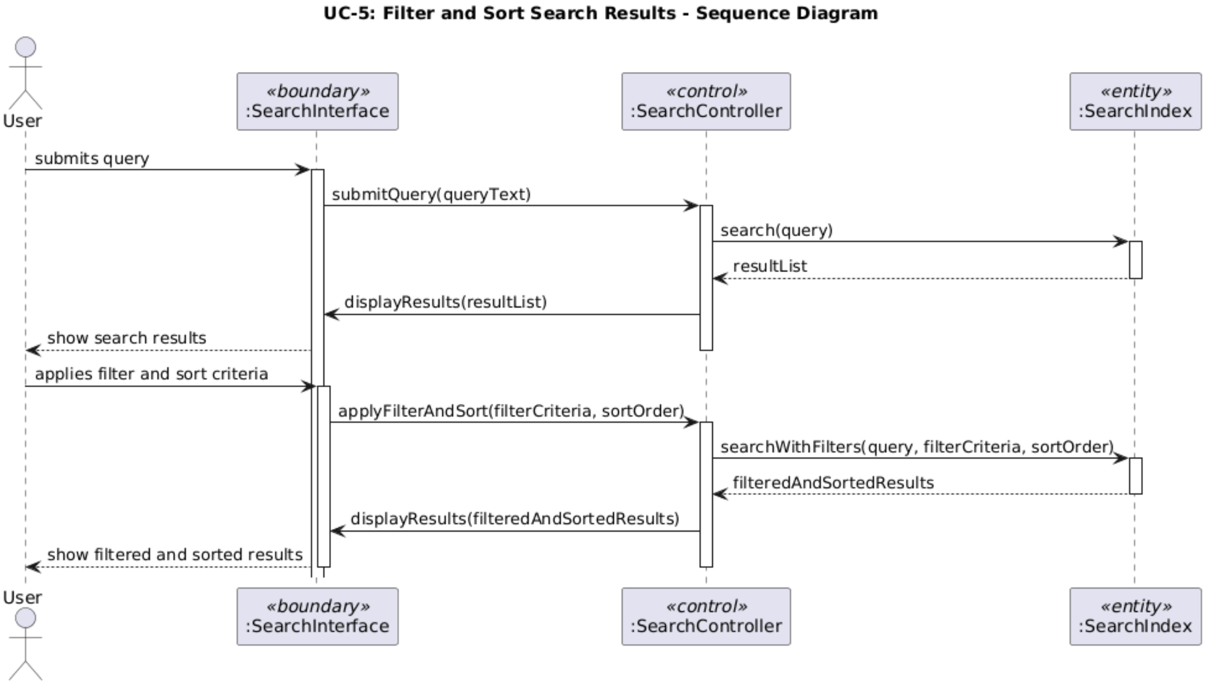
1. UC-1: Submit Search Query (REQ-1, REQ-2, REQ-3, REQ-4, REQ-10)
    - Complexity: Medium - involves query processing, autocorrection, result ranking, and pagination
    - Represents the primary user interaction and core search functionality

**UC-1: Submit Search Query - Sequence Diagram**



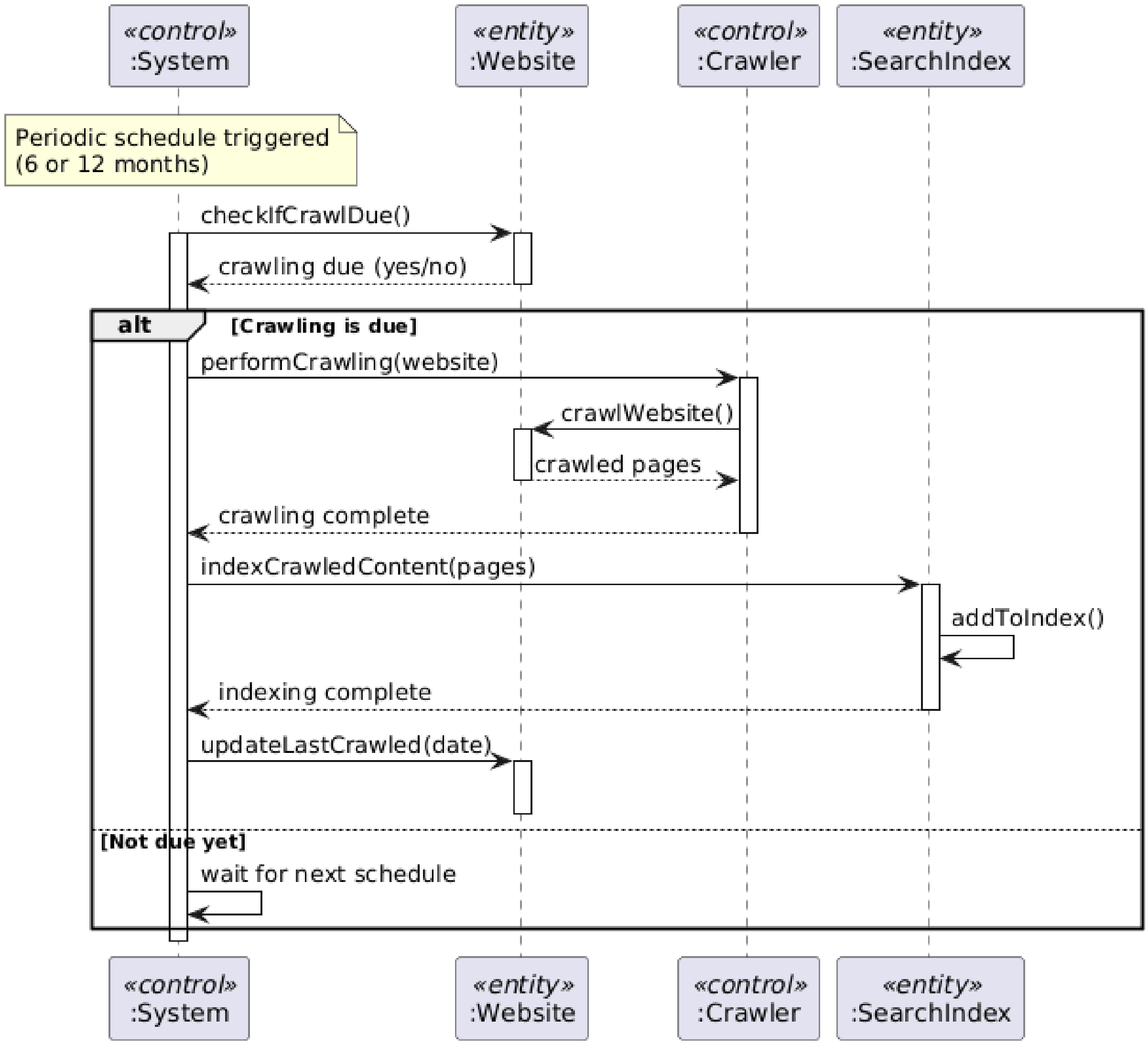
2. UC-5: Filter and Sort Search Results (REQ-5, REQ-8)
    - Complexity: High - involves re-querying the index with multiple criteria (language, date, relevance, last updated)

- Demonstrates the most complex user interaction with multiple parameters and dynamic result manipulation


UC-5: Filter and Sort Search Results - Sequence Diagram
User
«boundary» :SearchInterface
«control» :SearchController
«entity» :SearchIndex
submits query
submitQuery(queryText)
search(query)
resultList
displayResults(resultList)
show search results
applies filter and sort criteria
applyFilterAndSort(filterCriteria, sortOrder)
searchWithFilters(query, filterCriteria, sortOrder)
filteredAndSortedResults
displayResults(filteredAndSortedResults)
show filtered and sorted results
User
«boundary» :SearchInterface
«control» :SearchController
«entity» :SearchIndex

3. UC-13: Periodic Crawling (REQ-13, REQ-16, REQ-17, REQ-18, NFR-6, NFR-9)
   - Complexity: Highest - involves scheduled automation, website crawling, content indexing, and database updates
   - Represents the most complex system behaviour with automated scheduling, multi-component interaction (System → Website → Crawler → SearchIndex), and asynchronous processing

UC-13: Periodic Crawling - Sequence Diagram



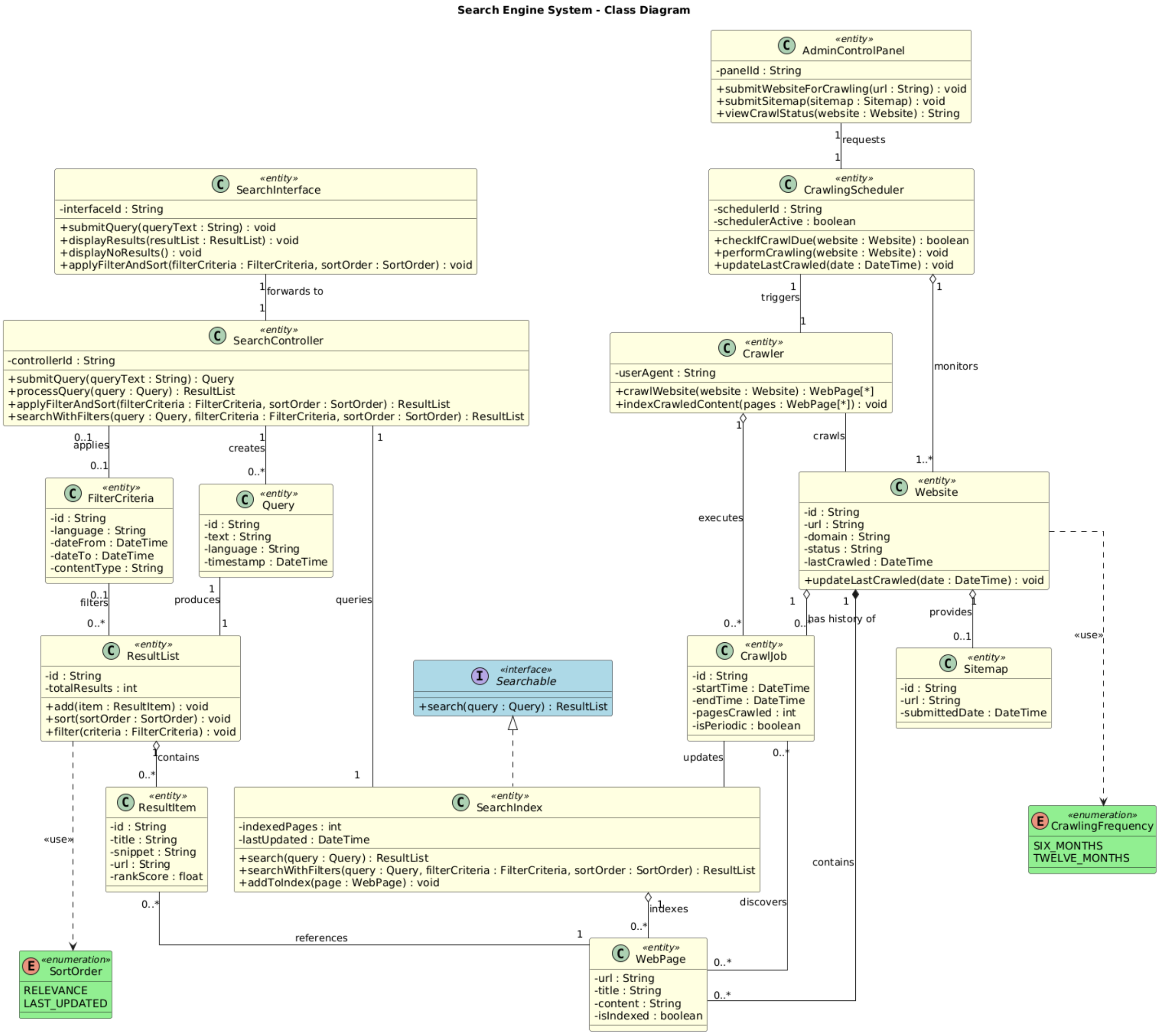
**Justification for Omitting the other use cases:**

These use cases either:

1. Represent **simpler operations** with fewer steps (UC-2, UC-3, UC-6, UC-7, UC-8)
2. Are **subsets or variants** of the selected use cases (UC-2 is part of UC-1, UC-12 is part of UC-13)
3. Involve **static configuration** rather than dynamic system behaviour (UC-11)
4. Have **similar complexity** to each other but lower than the selected ones (UC-9, UC-10)

## UML DIAGRAM

UML DIAGRAM OF THE WHOLE SYSTEM


Search Engine System - Class Diagram
«entity» AdminControlPanel
-panelId : String
+submitWebsiteForCrawling(url : String) : void
+submitSitemap(sitemap : Sitemap) : void
+viewCrawlStatus(website : Website) : String
requests
«entity» CrawlingScheduler
-schedulerId : String
-schedulerActive : boolean
+checkIfCrawlDue(website : Website) : boolean
+performCrawling(website : Website) : void
+updateLastCrawled(date : DateTime) : void
triggers
monitors
«entity» Crawler
-userAgent : String
+crawlWebsite(website : Website) : WebPage[*]
+indexCrawledContent(pages : WebPage[*]) : void
crawls
executes
«entity» Website
-id : String
-url : String
-domain : String
-status : String
-lastCrawled : DateTime
+updateLastCrawled(date : DateTime) : void
has history of
provides
«use»
«entity» Sitemap
-id : String
-url : String
-submittedDate : DateTime
«enumeration» CrawlingFrequency
SIX_MONTHS
TWELVE_MONTHS
«entity» CrawlJob
-id : String
-startTime : DateTime
-endTime : DateTime
-pagesCrawled : int
-isPeriodic : boolean
updates
contains
discovers
«entity» SearchInterface
-interfaceId : String
+submitQuery(queryText : String) : void
+displayResults(resultList : ResultList) : void
+displayNoResults() : void
+applyFilterAndSort(filterCriteria : FilterCriteria, sortOrder : SortOrder) : void
forwards to
«entity» SearchController
-controllerId : String
+submitQuery(queryText : String) : Query
+processQuery(query : Query) : ResultList
+applyFilterAndSort(filterCriteria : FilterCriteria, sortOrder : SortOrder) : ResultList
+searchWithFilters(query : Query, filterCriteria : FilterCriteria, sortOrder : SortOrder) : ResultList
applies
creates
queries
«entity» FilterCriteria
-id : String
-language : String
-dateFrom : DateTime
-dateTo : DateTime
-contentType : String
filters
«entity» Query
-id : String
-text : String
-language : String
-timestamp : DateTime
produces
«entity» ResultList
-id : String
-totalResults : int
+add(item : ResultItem) : void
+sort(sortOrder : SortOrder) : void
+filter(criteria : FilterCriteria) : void
contains
«use»
«interface» Searchable
+search(query : Query) : ResultList
«entity» ResultItem
-id : String
-title : String
-snippet : String
-url : String
-rankScore : float
«entity» SearchIndex
-indexedPages : int
-lastUpdated : DateTime
+search(query : Query) : ResultList
+searchWithFilters(query : Query, filterCriteria : FilterCriteria, sortOrder : SortOrder) : ResultList
+addToIndex(page : WebPage) : void
indexes
references
«enumeration» SortOrder
RELEVANCE
LAST_UPDATED
«entity» WebPage
-url : String
-title : String
-content : String
-isIndexed : boolean
1
0..1
0..*
1..*
0..*

## References

- **Title**: *Use Case Document for House Access Control*
  **Author**: Prof. Dr.-Ing. Kolja Eger
  **Version**: 1.0
  **Date**: 2025-11-07
  **Source**: SE_03_Eger_Use_Cases.pdf

- **Title**: *Requirements Engineering Process*
  **Author**: Prof. Dr.-Ing. Kolja Eger
  **Version**: 1.0

**Date**: 2025-10-30
**Source**: [SE_02b_EGR_Req_Eng (1).pdf] (file://mnt/data/SE_02b_EGR_Req_Eng.pdf)

- **Title**: *System Requirements Specification Template*
  **Author**: IEEE
  **Version**: IEEE Standard 830
  **Date**: 1998
  **Source**: IEEE Standard 830

- **Title**: *Sequence Diagrams*
  **Author**: Prof. Dr.-Ing. Kolja Eger
  **Source**: [SE_04b_Sequence_Diagrams.pdf]

**UML Modelling Standards**

- **Title**: Unified Modeling Language (UML) 2.0 Superstructure
  **Author**: Object Management Group (OMG)
  **Version**: 2.0
  **Date**: 2005
  **Source**: UML 2.0 Superstructure
  *This document provides comprehensive guidelines on using UML for modeling use cases, activities, and other system components, which you can refer to when creating your system models.*

**Performance and Reliability Guidelines**

- **Title**: *Software Engineering: A Practitioner's Approach*
  **Author**: Roger S. Pressman
  **Version**: 9th Edition
  **Date**: 2014
  **Source**: Software Engineering by Pressman
  *This reference can be used to discuss non-functional requirements like performance, reliability, and maintainability.*

**Interface Guidelines**

- ***Web Crawling and Indexing***

  *How Google Search Works: Crawling and Indexing Google Search Central Documentation Google LLC Source: https://developers.google.com/search/docs/crawling-indexing/overview*