Semantic Web and Ontology Project

Team Members

Ahmed Adel Ahmed **2100398**

Omar Othman Omar **2100153**

Dana Ahmed Mohamed **2100594**

Ahmed Ashraf Ali **2100390**

Eman Atif Hassan **2100623**

Essameldien Mohamed **2100658**

Semantic Search Engine Ontology Documentation

1. Overview

This project models a **semantic search engine** for books, authors, genres, publishers, and users using **Protégé**. The ontology represents knowledge about books, users, and their relationships, with **SWRL rules** for automated reasoning and **DL queries** for intelligent searches.

What Does This Project Do?

1. Represents core concepts:
    a. **Books**, **Authors**, **Genres**, **Publishers**, **Users**
2. Defines relationships between them using **Object Properties**:
    a. `hasAuthor` (Book → Author)
    b. `hasGenre` (Book → Genre)
    c. `likesGenre` (User → Genre)
    d. `recommendedBook` (User → Book)
3. Uses **SWRL Rules** to infer:
    a. Recommended books for users based on genre preferences.
    b. Classification of books as **New Releases** (published within the last 2 years).
4. Supports **DL Queries** for retrieving specific information.

2. Step-by-Step Implementation

Step 1: Create a New Ontology

1. Open **Protégé**.

2. Go to **File → New Ontology**.
3. Set an **IRI** (e.g., `http://www.semanticweb.org/hp/ontologies/2025/Search`).
4. Click **Next**, then **Finish**.

Step 2: Define Classes

1. Go to the **Classes** tab.
2. Right-click `owl:Thing` → **Add subclass**.
3. Create the following classes:
    a. `Book`
    b. `Author`
    c. `Genre`
    d. `Publisher`
    e. `User`
    f. `NewRelease` (subclass of `Book`)

Step 3: Define Object Properties

1. Go to the **Object Properties** tab.
2. Right-click `owl:topObjectProperty` → **Add subproperty**.
3. Define properties with **Domain** and **Range**:

Property	Domain	Range	Description

hasAuthor	Book	Author	Book is written by an author
hasGenre	Book	Genre	Book belongs to a genre
publishedBy	Book	Publisher	Book is published by a publisher
likesGenre	User	Genre	User prefers a genre
recommendedBook	User	Book	Book recommended for a user
publicationYear	Book	xsd:integer	Year the book was published

Step 4: Add Individuals

1. Go to the **Individuals** tab.
2. Select a class (e.g., Book) → **Add individual**.
3. Example individuals:

Authors

- Author1, Author2, Author3

Books

- Book1 (History, Author1, Publisher1)
- Book2 (Fantasy, Author2, Publisher1)
- Book3 (Science_Fiction, Author3, Publisher2)

Genres

- History, Fantasy, Science_Fiction

Publishers

- `Publisher1`, `Publisher2`

Users

- `User` (Likes: `Fantasy`, `History`)
- `User1` (Likes: `Fantasy`, `Science_Fiction`)

4. **Connect individuals** using **Object Property Assertions**:
    a. `Book1` → `hasAuthor` → `Author1`
    b. `Book1` → `hasGenre` → `History`
    c. `User` → `likesGenre` → `Fantasy`

Step 5: Add SWRL Rules

1. Go to the **SWRLTab**.
2. Click **+** to add a rule.

Rule 1: Book Recommendation

```
swrl
User(?u) ^ likesGenre(?u, ?g) ^ Book(?b) ^ hasGenre(?b, ?g)
→ recommendedBook(?u, ?b)
```

Effect:

- Recommends books to users based on their preferred genres.

Rule 2: New Release Classification

```
swrl
```

```
Book(?b) ^ publicationYear(?b, ?y) ^ subtract(2025, ?y, ?d) ^ lessThan(?d, 2)
→ NewRelease(?b)
```

Effect:

- Books published in **2024 or 2025** are classified as `NewRelease`.

Step 6: Run the Reasoner

1. Go to **Reasoner → ELK → Start reasoner**.
2. Let the system infer new relationships (e.g., `recommendedBook`).

Step 7: Query the Ontology (DL Queries)

1. Go to the **DL Query** tab.
2. Example queries:

Query 1: Find All Recommended Books for a User

```
dl
recommendedBook value User
```

Result:

- Book1, Book2

Query 2: Find All Fantasy Books

```
dl
hasGenre value Fantasy
```

Result:

- Book2