SEMANTIC WEB – PROJECT

TITLE: Train Search Engine Project

- Mahima S

[2022115018]

1. Project Overview:

The "Train Search Engine" is a web-based application designed to allow users to search for and retrieve information related to trains, routes, and stations from an RDF-based ontology. The application enables users to query data on train operators, types, capacities, route details, station locations, and more. The search engine leverages the powerful querying capabilities of SPARQL to retrieve relevant results from an RDF graph.

2. Objectives:

- **Create a search engine to allow users to search and retrieve train-related information from an RDF-based ontology.**
- **Design and implement a user-friendly interface for entering search queries and displaying results.**

- **Ensure the application supports querying different categories (Train, Route, and Station) with dynamic results presentation.**

3. Technologies Used:

- **Python (Flask Framework): Used for the backend to handle HTTP requests, process form data, and integrate with the RDF data.**
- **SPARQL: Used to query the RDF ontology for relevant information.**
- **RDFLib: Python library to parse and work with RDF data.**
- **HTML/CSS: Used for designing the frontend of the application. The layout is clean, user-friendly, and responsive.**
- **RDF/OWL Ontology: The data model for the train, route, and station information is represented using RDF (Resource Description Framework) and OWL (Web Ontology Language) standards.**

4. Functional Components:

1. **Backend (Flask Application):**
    - **The Flask app serves as the core of the application, handling both GET and POST requests.**
    - **The user selects a search type (Train, Route, or Station) and provides a search keyword.**
    - **The application constructs a SPARQL query based on the search type and value.**
    - **It then queries the RDF ontology (stored as an OWL file) to retrieve the relevant information.**
    - **The results are processed and passed to the frontend for display.**

2. **SPARQL Queries:**
    - **Queries are dynamically constructed based on user input.**
    - **For each search type (Train, Route, Station), different SPARQL queries are used to fetch the necessary details from the ontology.**

        - **Train Query: Fetches train operator, type, and capacity.**
        - **Route Query: Fetches route name, departure time, arrival time, and description.**
        - **Station Query: Fetches station name and location.**

3. **Frontend (HTML & CSS):**
    - **The frontend consists of a search form where users can input their query.**
    - **A responsive layout ensures that the application works well on both desktop and mobile devices.**
    - **The results are displayed in a structured format, with each type of result (Train, Route, or Station) showing relevant fields in a clean and organized manner.**

5. RDF Ontology Example:

The ontology used in the project represents the following classes:

- **Train: Represents train details, including operator, type, and capacity.**

- **Route: Represents train routes with properties like route name, departure time, arrival time, and route description.**
- **Station: Represents train stations, including station name and location.**

6. Detailed Breakdown of Search and Results:

1. **Train Search:**
   - **Query: Searches for trains based on the operator name or type.**
   - **Results: Displays the operator, type, capacity, and related route information.**
2. **Route Search:**
   - **Query: Searches for routes based on the route name.**
   - **Results: Displays route details including route name, departure time, arrival time, and description.**
3. **Station Search:**

    - Query: Searches for stations based on the station name.
    - Results: Displays the station name, location, and associated train services.

7. Challenges Encountered:

- Data Representation: Structuring the RDF data to fit the real-world scenario of trains, routes, and stations while maintaining clarity and reusability.
- Query Complexity: Constructing dynamic SPARQL queries based on user input, ensuring that results are accurate and relevant.
- User Interface: Making sure that the frontend was intuitive, and results were easy to read and understand.

8. Conclusion:

The "Train Search Engine" project demonstrates the effective use of semantic web technologies, such as RDF and SPARQL, to build an intelligent search engine for train-related data. The combination of Flask for backend

services and SPARQL for querying an RDF-based ontology provides a powerful platform for querying and displaying train, route, and station details. The project is flexible and can be expanded further to include additional types of data and improve the user interface.

CODE :

App.py

```
from flask import Flask, render_template, request

from rdflib import Graph, Namespace


app = Flask(__name__)


# Load the ontology

g = Graph()

ontology_path = "train_ontology.owl"

g.parse(ontology_path, format="xml")


# Define namespaces

EX = Namespace("http://example.org/train#")

g.bind("ex", EX)
```

```
# Function to query the ontology

def search_ontology(search_type, search_value):

    if search_type == "train":

        query = f"""

        PREFIX ex: <http://example.org/train#>

        SELECT ?operator ?type ?capacity ?route

        WHERE {{

            ?train a ex:Train ;

                   ex:trainOperator ?operator ;

                   ex:trainType ?type ;

                   ex:hasTrainCapacity ?capacity ;

                   ex:operatesOnRoute ?route .

            FILTER regex(str(?train), "{search_value}", "i")

        }}

        """

    elif search_type == "route":

        query = f"""

        PREFIX ex: <http://example.org/train#>

        SELECT ?routeName ?departure ?arrival ?description ?startStation
?endStation

        WHERE {{
```

```
        ?route a ex:Route ;
               ex:routeName ?routeName ;
               ex:departureTime ?departure ;
               ex:arrivalTime ?arrival ;
               ex:routeDescription ?description ;
               ex:hasStartStation ?startStation ;
               ex:hasEndStation ?endStation .
        FILTER regex(?routeName, "{search_value}", "i")
    }}
    """
elif search_type == "station":
    query = f"""
    PREFIX ex: <http://example.org/train#>
    SELECT ?stationName ?stationLocation ?servedTrain
    WHERE {{
        ?station a ex:Station ;
                 ex:stationName ?stationName ;
                 ex:stationLocation ?stationLocation ;
                 ex:servesTrain ?servedTrain .
        FILTER regex(?stationName, "{search_value}", "i")
    }}
    """
    else:
        return []


    results = g.query(query)
    return results


@app.route("/", methods=["GET", "POST"])
def index():
    results = None
    if request.method == "POST":
        search_type = request.form.get("search_type")
        search_value = request.form.get("search_value")
        results = search_ontology(search_type, search_value)
    return render_template("index.html", results=results,
search_type=request.form.get("search_type", ""))


if __name__ == "__main__":
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Train Search Engine</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f9;
            margin: 0;
            padding: 0;
        }
        .container {
            max-width: 800px;
            margin: 50px auto;
            background: #ffffff;
            padding: 20px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            border-radius: 10px;
        }
        h1 {
            text-align: center;
    color: #333;
}
form {
    display: flex;
    flex-direction: column;
    gap: 15px;
}
label {
    font-weight: bold;
    color: #555;
}
input, select, button {
    padding: 10px;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 5px;
}
button {
    background-color: #007BFF;
    color: #fff;
    cursor: pointer;
    border: none;
    transition: background-color 0.3s;
}
button:hover {
    background-color: #0056b3;
}
.results {
    margin-top: 20px;
}
.results h2 {
    text-align: center;
    color: #444;
}
.result-item {
    padding: 15px;
    background: #f9f9f9;
    border: 1px solid #eaeaea;
    margin-bottom: 15px;
    border-radius: 5px;
}
.result-item span {
    font-weight: bold;
    color: #007BFF;
```

```
        }
        .result-label {
            font-weight: bold;
            color: #555;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Train Search Engine</h1>
        <form method="POST">
            <label for="search_type">Search Type:</label>
            <select name="search_type" id="search_type" required>
                <option value="train">Train</option>
                <option value="route">Route</option>
                <option value="station">Station</option>
            </select>
            <label for="search_value">Search Value:</label>
            <input type="text" id="search_value" name="search_value"
placeholder="Enter search keyword..." required>
            <button type="submit">Search</button>
        </form>
        {% if results %}
        <div class="results">
            <h2>Results:</h2>
            {% if search_type == "train" %}
                {% for row in results %}
                <div class="result-item">
                    <p><span class="result-label">Operator:</span> {{
row.operator }}</p>
                    <p><span class="result-label">Type:</span> {{ row.type
}}</p>
                    <p><span class="result-label">Capacity:</span> {{
row.capacity }}</p>
                    <p><span class="result-label">Route:</span> {{ row.route
}}</p>
                </div>
                {% endfor %}
            {% elif search_type == "route" %}
                {% for row in results %}
                <div class="result-item">
                    <p><span class="result-label">Route Name:</span> {{
row.routeName }}</p>
                    <p><span class="result-label">Departure Time:</span> {{
row.departure }}</p>
```

```
                <p><span class="result-label">Arrival Time:</span> {{
row.arrival }}</p>
                <p><span class="result-label">Description:</span> {{
row.description }}</p>
                <p><span class="result-label">Start Station:</span> {{
row.startStation }}</p>
                <p><span class="result-label">End Station:</span> {{
row.endStation }}</p>
            </div>
            {% endfor %}
        {% elif search_type == "station" %}
            {% for row in results %}
            <div class="result-item">
                <p><span class="result-label">Station Name:</span> {{
row.stationName }}</p>
                <p><span class="result-label">Location:</span> {{
row.stationLocation }}</p>
                <p><span class="result-label">Served Train:</span> {{
row.servedTrain }}</p>
            </div>
            {% endfor %}
        {% endif %}
    </div>
```

```
        {% endif %}
    </div>
</body>
</html>
```

train_ontology.owl

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://example.org/train#">

    <!-- Train Instances -->
    <rdf:Description rdf:about="http://example.org/train#Express123">
        <rdf:type rdf:resource="http://example.org/train#Train"/>
        <ex:trainOperator>National Rail</ex:trainOperator>
        <ex:trainType>Express</ex:trainType>
        <ex:hasTrainCapacity
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">250</ex:ha
sTrainCapacity>
        <ex:operatesOnRoute
rdf:resource="http://example.org/train#BlueLine"/>
    </rdf:Description>

    <rdf:Description rdf:about="http://example.org/train#Rapid789">
        <rdf:type rdf:resource="http://example.org/train#Train"/>
        <ex:trainOperator>City Transit</ex:trainOperator>

        <ex:trainType>Rapid</ex:trainType>

        <ex:hasTrainCapacity
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">180</ex:ha
sTrainCapacity>

        <ex:operatesOnRoute
rdf:resource="http://example.org/train#GreenLine"/>

    </rdf:Description>


    <!-- Route Instances -->

    <rdf:Description rdf:about="http://example.org/train#BlueLine">

        <rdf:type rdf:resource="http://example.org/train#Route"/>

        <ex:routeName>BlueLine</ex:routeName>

        <ex:departureTime>08:30 AM</ex:departureTime>

        <ex:arrivalTime>02:15 PM</ex:arrivalTime>

        <ex:routeDescription>Coastal route covering scenic
locations</ex:routeDescription>

        <ex:hasStartStation
rdf:resource="http://example.org/train#GrandCentral"/>

        <ex:hasEndStation
rdf:resource="http://example.org/train#CentralPark"/>

    </rdf:Description>
    <rdf:Description rdf:about="http://example.org/train#GreenLine">
        <rdf:type rdf:resource="http://example.org/train#Route"/>
        <ex:routeName>GreenLine</ex:routeName>
        <ex:departureTime>09:00 AM</ex:departureTime>
        <ex:arrivalTime>01:30 PM</ex:arrivalTime>
        <ex:routeDescription>Mountain route passing through hilly
regions</ex:routeDescription>
        <ex:hasStartStation
rdf:resource="http://example.org/train#DowntownStation"/>
        <ex:hasEndStation
rdf:resource="http://example.org/train#HighlandPoint"/>
    </rdf:Description>

    <!-- Station Instances -->
    <rdf:Description rdf:about="http://example.org/train#GrandCentral">
        <rdf:type rdf:resource="http://example.org/train#Station"/>
        <ex:stationName>GrandCentral</ex:stationName>
        <ex:stationLocation>Downtown City</ex:stationLocation>
        <ex:servesTrain
rdf:resource="http://example.org/train#Express123"/>
    </rdf:Description>

    <rdf:Description rdf:about="http://example.org/train#CentralPark">
```

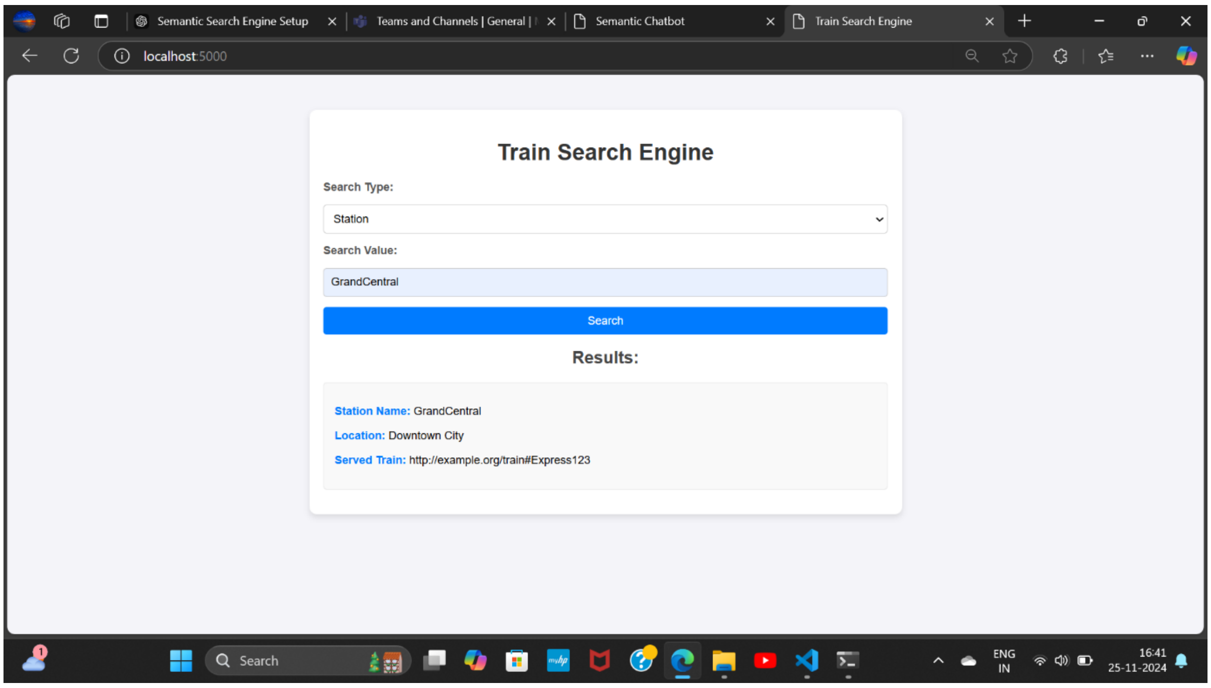
```
        <rdf:type rdf:resource="http://example.org/train#Station"/>
        <ex:stationName>CentralPark</ex:stationName>
        <ex:stationLocation>City Outskirts</ex:stationLocation>
        <ex:servesTrain
rdf:resource="http://example.org/train#Express123"/>
    </rdf:Description>

    <rdf:Description
rdf:about="http://example.org/train#DowntownStation">
        <rdf:type rdf:resource="http://example.org/train#Station"/>
        <ex:stationName>DowntownStation</ex:stationName>
        <ex:stationLocation>City Center</ex:stationLocation>
        <ex:servesTrain rdf:resource="http://example.org/train#Rapid789"/>
    </rdf:Description>

    <rdf:Description rdf:about="http://example.org/train#HighlandPoint">
        <rdf:type rdf:resource="http://example.org/train#Station"/>
        <ex:stationName>HighlandPoint</ex:stationName>
        <ex:stationLocation>Hill Station</ex:stationLocation>
        <ex:servesTrain rdf:resource="http://example.org/train#Rapid789"/>
    </rdf:Description>
```

```
</rdf:RDF>
```

Output ScreenShots:



Train Search Engine

Search Type: Station

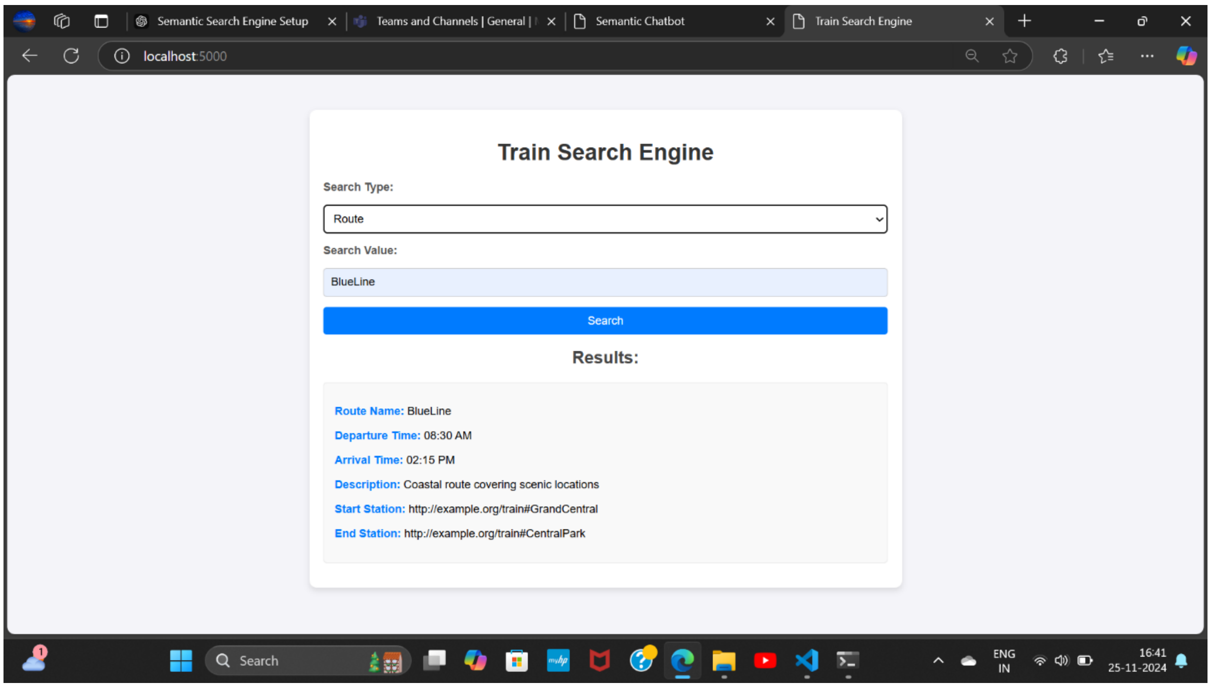
Search Value: GrandCentral

Results:

Station Name: GrandCentral

Location: Downtown City

Served Train: http://example.org/train#Express123



Train Search Engine

Search Type:

Route

Search Value:

BlueLine

Search

Results:

Route Name: BlueLine

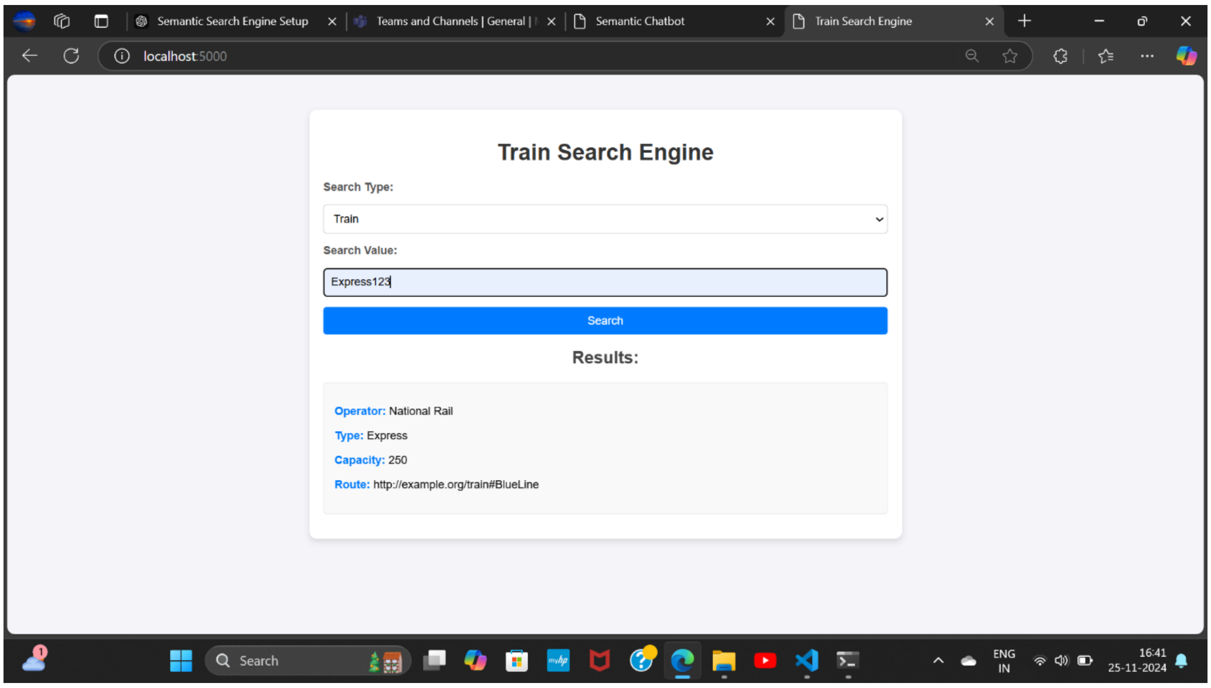
Departure Time: 08:30 AM

Arrival Time: 02:15 PM

Description: Coastal route covering scenic locations

Start Station: http://example.org/train#GrandCentral

End Station: http://example.org/train#CentralPark



Train Search Engine

Search Type:

Train

Search Value:

Express123

Search

Results:

Operator: National Rail

Type: Express

Capacity: 250

Route: http://example.org/train#BlueLine