**MOD004979**

**SID: 2022785**

Semantic data technologies

Table of Contents

[Table of figures 2](#_Toc89851128)

[1. Introduction 3](#_Toc89851129)

[2. Concept/Aim 4](#_Toc89851130)

[3. Ontology Design 5](#_Toc89851131)

[4. Implementation 10](#_Toc89851132)

[4.1. Technology 10](#_Toc89851133)

[4.2. Application Structure 10](#_Toc89851134)

[4.3. User Interface 12](#_Toc89851135)

[5. Evaluation and Use 15](#_Toc89851136)

[5.1. Basic Search Query 15](#_Toc89851137)

[5.2. Advance Search Query 16](#_Toc89851138)

[5.3. Get Departments Query 17](#_Toc89851139)

[5.4. Get Employment Type Query 18](#_Toc89851140)

[5.5. Get Department Roles Query 18](#_Toc89851141)

[5.6. Get Leaves Query 19](#_Toc89851142)

[5.7. Get Employees Query 19](#_Toc89851143)

[5.8. Get Branches Query 20](#_Toc89851144)

[6. Critical Reflection 21](#_Toc89851145)

[7. Conclusion 22](#_Toc89851146)

[8. Reference 23](#_Toc89851147)

# Table of figures

[Figure 1 Employee Ontology Metrics 5](#_Toc89807072)

[Figure 2 List of Classes and Sub-Classes in the Ontology 6](#_Toc89807073)

[Figure 3 Development and Accountant and Finance Roles Class Description 7](#_Toc89807074)

[Figure 4 Employee class description 7](#_Toc89807075)

[Figure 5 Ontology Object Properties 8](#_Toc89807076)

[Figure 6 'is on' object property characteristics, domain, and range 8](#_Toc89807077)

[Figure 7 Ontology Data Properties 9](#_Toc89807078)

[Figure 8 Employee Individual data 9](#_Toc89807079)

[Figure 9 Employee Management System - Homepage 12](#_Toc89807080)

[Figure 10 Employee Management System - Description Page 13](#_Toc89807081)

[Figure 11 Basic Search Filter Query 15](#_Toc89807082)

[Figure 12 Basic Search Query Results 15](#_Toc89807083)

[Figure 13 Advance Filter Search Results 17](#_Toc89807084)

[Figure 14 Get Departments Query and Results 17](#_Toc89807085)

[Figure 15 Get Employment Type Query and Results 18](#_Toc89807086)

[Figure 16 Get Department Roles Query and Results 18](#_Toc89807087)

[Figure 17 Get Leaves Query and Results 19](#_Toc89807088)

[Figure 18 Get Employees Query and Results 19](#_Toc89807089)

[Figure 19 Get Branches Query and Results 20](#_Toc89807090)

# Introduction

The term Semantic Web is first introduced by Tim Berners-Lee. It is the framework that allows applications to share and reuse data that is currently available on the internet. This technology helps organizations to create and handle data using rules created by technologies such as RDF, SPARQL, and Ontology. The purpose of this project is to create a Semantic application that uses the concepts of ontology and RDF. The employee management system explained in this report will help organizations or companies to search for employees who are currently working. The report is further divided into five sections, i.e, Concept/Aim of the application, Ontology design that is used by the employee management system, UI interface of the application, queries used in the application, and the critical reflection of the application. And finally, the report concludes with the conclusion section. Below is the list of software used while developing the application.

**SYSTEM REQUIREMENTS**

* Apache Jena Fuseki Server v4.2.0 (Apache Jena - Apache Jena Fuseki, 2021).
* PyCharm Community Editor 2021.2.3 (JetBrains - PyCharm, 2021).
* Google Chrome 96.0.4664.45 (Google.com, 2021).
* Microsoft Excel 2103 (16.0.13901.20400) (Microsoft Excel Spreadsheet Software | Microsoft 365, 2021).

# Concept/Aim

Currently, there are many employee management systems available on the internet that manages employees’ data using different technologies and databases. With the available roles and departments available in the software industry, the application can make use of semantic web technologies to store and retrieve the knowledge that is related to the roles, departments, and employees. The existing HR ontology contains knowledge that is limited to employees working in IT software fields and is limited to a few departments and roles (Naz, Ali, and Siddiqui, 2019). The proposed ontology is similar to existing ontology but focuses more on departments and the roles available in the ontology are categorized department-wise. This ontology contains more classes and properties than the existing one.

The main aim of the employee management system is to help companies to organize the existing employee data. Using this application, the administrators and HR’s working in the company can keep track of employees working on specific projects or for employees who are on annual or sick leave. This application will also allow administrators to see a list of employees working in a particular branch or department.

The user interfaces implemented in the application will allow administrators to search for an employee using different dropdowns and text-based searches. Overall, the entire application consists of two pages where one page contains all information about all roles, departments, employees that are currently available in the database. And the second page in the application will give detailed information about the selected employee. The homepage in the application consists of an employee’s table and filter section. This filter section contains the list of dropdown filters and a text-based search field. The administrator can use the search field to search for the employee using their names or on the other hand, the administrator can filter the employee’s data with the help of options listed in those dropdown fields. Administrators/HR can even perform an advance search by making use of both dropdowns and input fields. The employee details page in the application will show the entire information of the selected employee that is available in the database.

# Ontology Design

The ontology named ‘Employee Management System’ used in this application is created using open-source ontology building software known as Protégé (protégé, 2021). In the employee ontology, there is a total of 2415 axioms, 1999 logical axioms, 86 class, 15 object properties, 20 data properties, and over 175 individuals are defined. There are a total of 11 classes in this ontology that are considered as the disjoint classes.

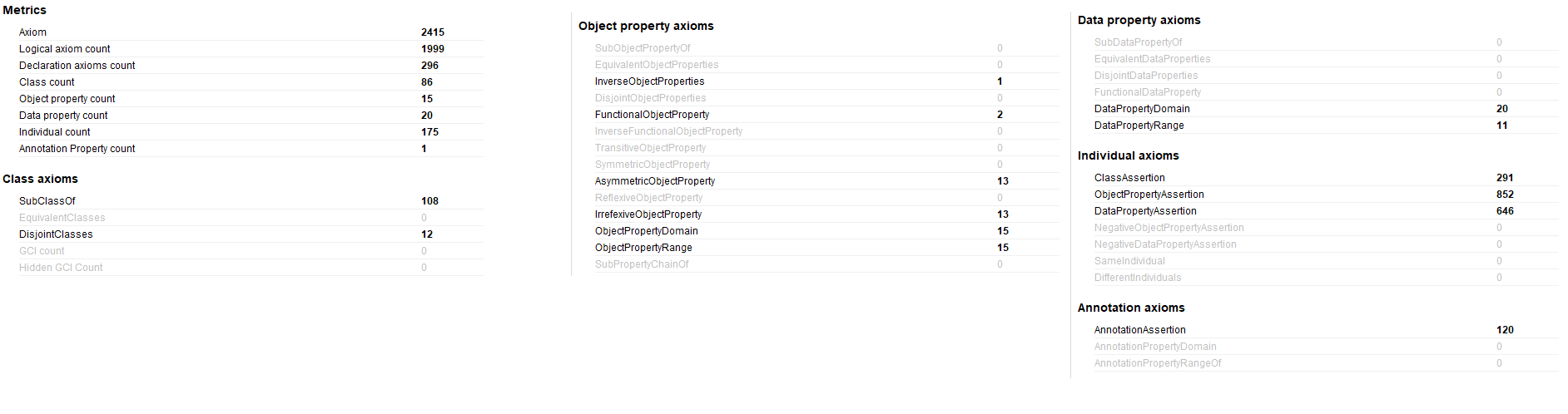


Figure 1 Employee Ontology Metrics

The employee Ontology metrics above show a total number of data properties and object properties axioms available. Out of 86 classes, 11 classes are considered the main classes in this ontology. Each of these 11 classes has a list of defined subclasses. Below is the list of main classes defined.

* Branch
* Employment Type
* Employees
* Departments
* Roles
* Projects
* Education
* Certifications
* Country
* Level
* Leaves

Few of these main classes are further categorized into sub-classes based on their functionality. As per the real-world scenario, few classes are considered as the disjoint while others are not. For example, the departments class further consists of 6 sub-classes and each of these sub-classes is defined as disjoint classes whereas the sub-classes available in the education class are not considered as the disjoint classes. This is because an employee can have multiple qualifications, but the employee can only be able to work in one department. Similarly, the sub-classed defined in the roles class are categorized based on the departments specified in the department class. So, the development roles subclass defined in the roles class can contain only roles that belong to the development department. Classes like Branch, Leaves, Employment Type, Level also contain disjoint classes.

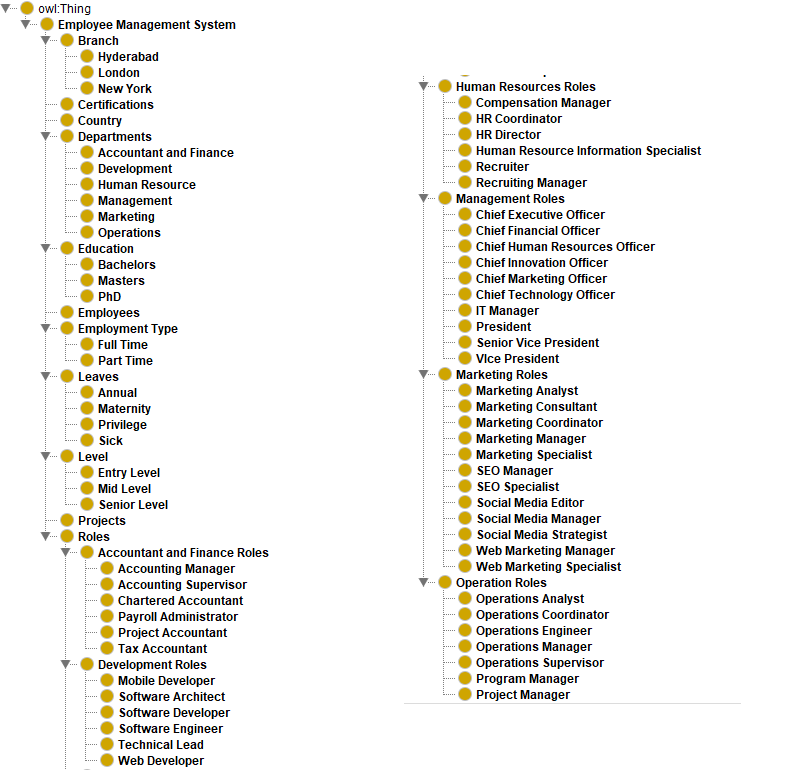


Figure 2 List of Classes and Sub-Classes in the Ontology

Based on the functionality and the purpose of each class, few restrictions have been set to certain classes. For example, in below figure 3, the development class can contain only roles that belong to the development department. So, the condition “‘has role’ only ‘Development Roles’” has been specified in the development class description whereas the accountant and finance roles can belong only to the Accountant and Finance Department. So, the condition “’ belongs to’ only ‘Accountant and Finance’ is specified.

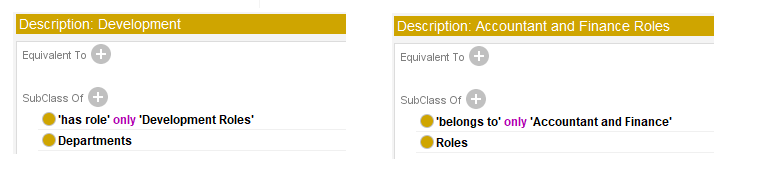


Figure 3 Development and Accountant and Finance Roles Class Description

The employee class defined in the ontology is connected to many other different classes. So, the employee class contains many conditions when compared with other classes. Conditions like “certified as’ some Certifications”, “‘has completed’ some Education”, “’ works on’ some Projects” will allow individuals of type employee to have more than one relationship with different individuals of the same type whereas the conditions like “’ has experienced’ max 1 Level”, “’ is on’ max 1 Leaves”, “is a’ max 1 ‘Employment Type” will restrict the employee individuals to have not more than one relationship with other individuals. Figure 4 will show the list of conditions specified to the Employee class.

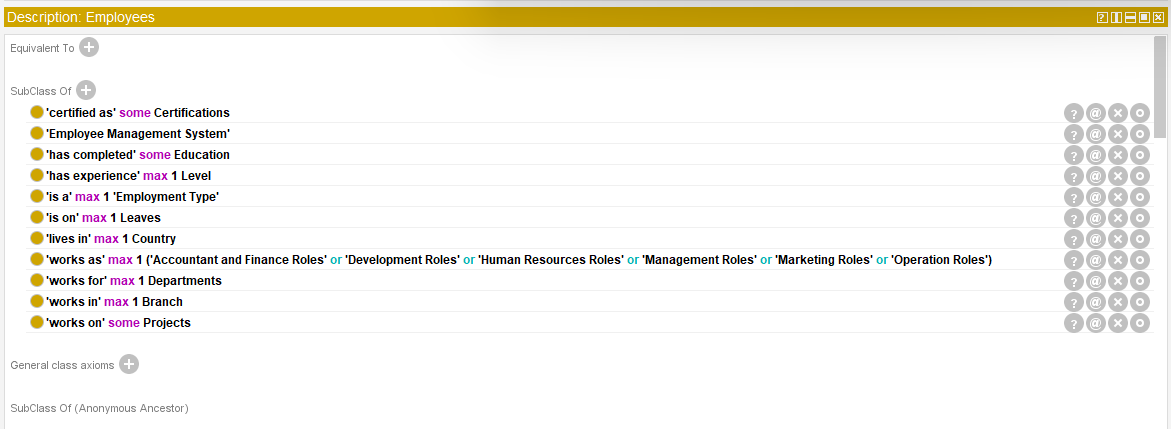


Figure 4 Employee class description

The main functionality of the object properties is to maintain the relationship between two different individuals whereas the data properties in the ontology connect individuals to some form of attribute values like string, integer, float, and date, etc. There is a total of 15 object properties. Figure 5 shows the list of object properties available in the ontology.

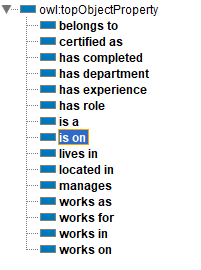


Figure 5 Ontology Object Properties

Each object property defined in the ontology has different functionality when compared to others. For example, the ‘belongs to’ object property links the individual of type Roles to the individual of type Department. And this forms a relationship as ‘Roles belongs to Departments’. Similarly, the ‘works on’ property links individual of type Employee to the individual of type Projects creating ‘Employee works on Projects’ relationship.

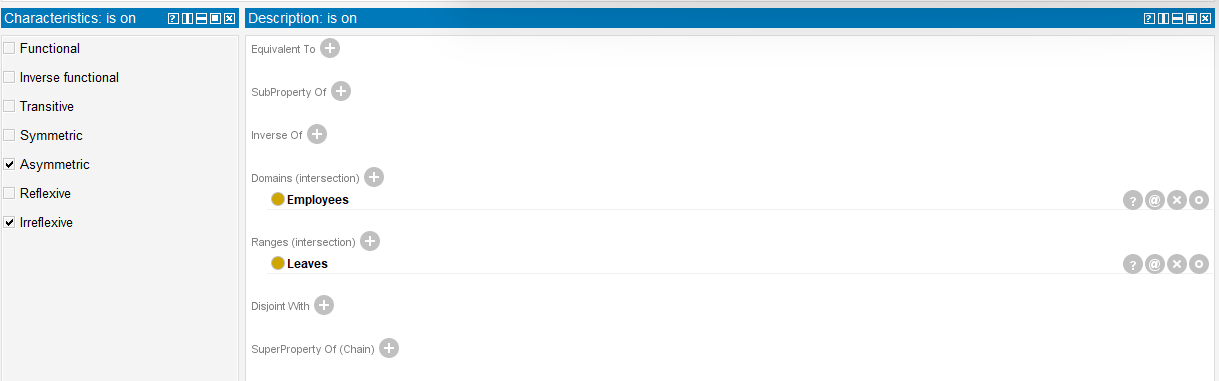


Figure 6 'is on' object property characteristics, domain, and range

Each object property can have a list of characteristics. The main purpose of the characteristics of data property and object property is to tell the ontology how the relationship between two individuals works. Most of the object properties defined in the ontology have ‘Asymmetric’ and ‘Irreflexive’ characteristics. If the property has ‘Asymmetric’ characteristics, then the two individuals that are linked with that property cannot form a circular relationship whereas the ‘Irreflexive’ characteristic restricts the individual to link itself via that property. Out of 15 object properties, only two properties ‘has role’ and ‘belongs to’ are declared as the inverse properties.

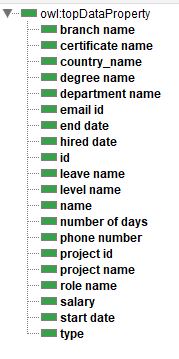


Figure 7 Ontology Data Properties

In the employee ontology, there is a total of 20 data properties available, few data properties have different domains when compared with others. The main purpose of these data properties is to store more information about the individuals. For example, data properties like name, phone number, id, email id, hired date contains more information about the individual of type employee. Most of the data properties have a range as ‘xsd:string’. Meaning those data properties can store data of type string.

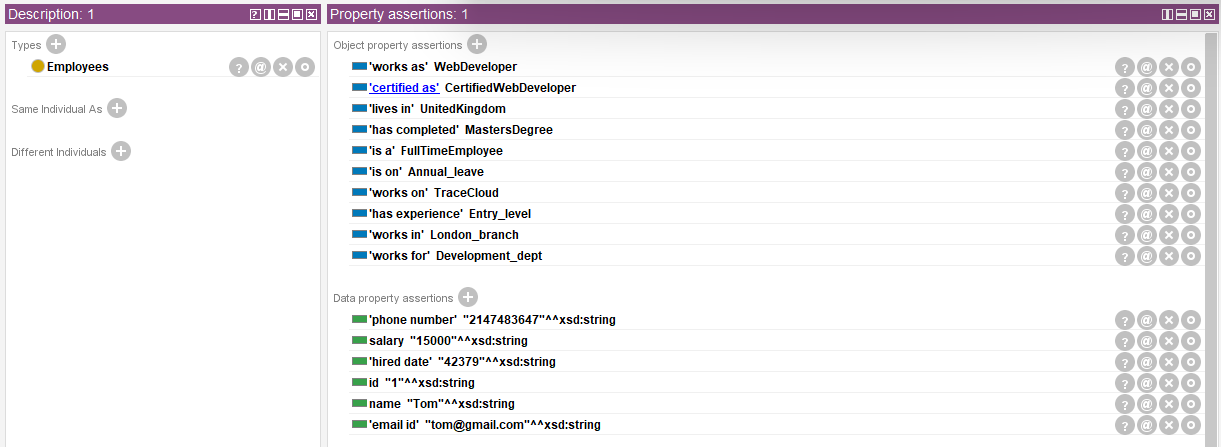


Figure 8 Employee Individual data

The above figure shows the information about the individual of type employee. Using object properties, individuals of different types like a branch, certifications, country are connected to the employee individual and with the help of data properties, more information about the employee is linked to the individual.

# Implementation

## Technology

The entire application is developed using the Django framework. Django is a high-level python web framework used to develop small, medium, and large-scale web applications. Scalability and Security are the two major features of the Django framework. These features by default will help the developer while developing the application. The front-end of the application is designed and developed using HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), Ajax (Asynchronous JavaScript and XML), and the back end of the application is developed using Python programming language. This application also makes use of the MVT (Model-View-Template) concept. Each component in the MVT is responsible for handling the functionality of the application. The Model component contains back-end code that is responsible for retrieving the data from the database that is stored on the Apache Jena Server and sending the data to the View. The View component in the MVT architecture is responsible for sending the user request to the Model and returning the response to the Template component. The Template component controls the user interface of the application. This folder contains all the files that are related to the user interface.

The functions declared in the model use the SPARQLWRAPPER library that is installed in the application. This SPARLWRAPPER library is a python wrapper around SPARQL services that executes the SPARQL queries and returns the results in a more manageable format like JSON. With the help of pre-defined functions in the SPARQLWRAPPER library, the Django application can connect with the fuseki server using the SPARQL endpoint.

## Application Structure

The employee management system contains three main folders, i.e., static, EmployeeSystem and templates folders. The static folder in the application contains all files related to the front-end design and functionality of an application, i.e., CSS and JavaScript files. The templates folder is responsible for storing all available HTML pages of the application, i.e., Homepage.html and Description.html. The EmployeeSystem folder is the home for all python files that resides in the application. Below is the list of python files stored in the EmployeeSystem folder.

* settings.py
* urls.py
* views.py
* models.py

Each python file is responsible for delivering separate functionality of the entire application. The code in the ‘settings.py’ file is responsible for the Django web application to run. The file stores the configuration values related to the database settings, location static files, and information about the installed apps. The ‘urls.py’ file in the EmployeeSystem application is responsible for handling requests made by the end-users. This urls.py contains the ‘urlpatterns’ tuple that stores the mapping information about the URLs of the applications and the functions in the views.py file. The functions defined in the ‘views.py’ file are known as the view functions. The job of these functions is to handle the user request and return the response or redirect the user as per the request. Below is the list of functions defined in the ‘views.py’ file.

* index
* description
* search\_results
* advance\_search

The ‘index’ function is responsible for loading the homepage of the application whenever the user visits the application. This function gets all the subclasses and individuals of type employees defined in the ontology and populates those values as options in the advance search filters and employees in the table, respectively. By default, the application shows all the employee's types of individuals defined in the ontology. The ‘description’ function redirects the user to the description page. Before redirecting the user, the function gets the entire details of the selected employee and then redirects to the description page. The ‘search\_results’ and ‘advance\_search’ functions will get executed as per the user filter preferences and return the data to the templates.

The ‘models.py’ file in the application stores code that is responsible for retrieving the data from the Apache Jena Fuseki server. The ‘SPARQLMODEL’ class in this file contains functions listed below.

* get\_employees – this function returns all employees individuals available in the employee management system ontology.
* get\_roles – this function returns all roles available in the employee management system ontology.
* get\_departments – this function returns all departments available as the subclasses in the department class.
* get\_branches - this function returns all branches available as the subclasses in the branch class.
* get\_levels - this function returns all levels available as the subclasses in the level class.
* get\_employment\_type - this function returns all employment type subclasses available in the employment class.
* get\_education - this function returns all education subclasses available in the education class.
* get\_leaves - this function returns all leaves subclasses available in the leaves class.
* get\_projects - this function returns all individuals of type projects available in the employee system ontology.
* get\_department\_roles – this function returns all roles linked with the selected department.
* search\_employees – this function returns all employee individuals if the value passed from the view exists in the employee’s name value.
* get\_advance\_search – this function returns all employee individuals that match with the filters passed from the view function.
* employee\_details – this function returns entire information or object properties and data properties that are linked with the selected employee.

## User Interface

The user interface will many features and options that will easily overwhelm the end-user and makes the end-user focus on key features of the application. To avoid the issue related to the user experience, the development process followed the concepts like SUI (Simplified User Interface) and PEOU (Perceived Ease of Use). Also, while designing the user interface for this application, a few methods and factors are taken into consideration (Chandra and Guntupalli, n.d.).

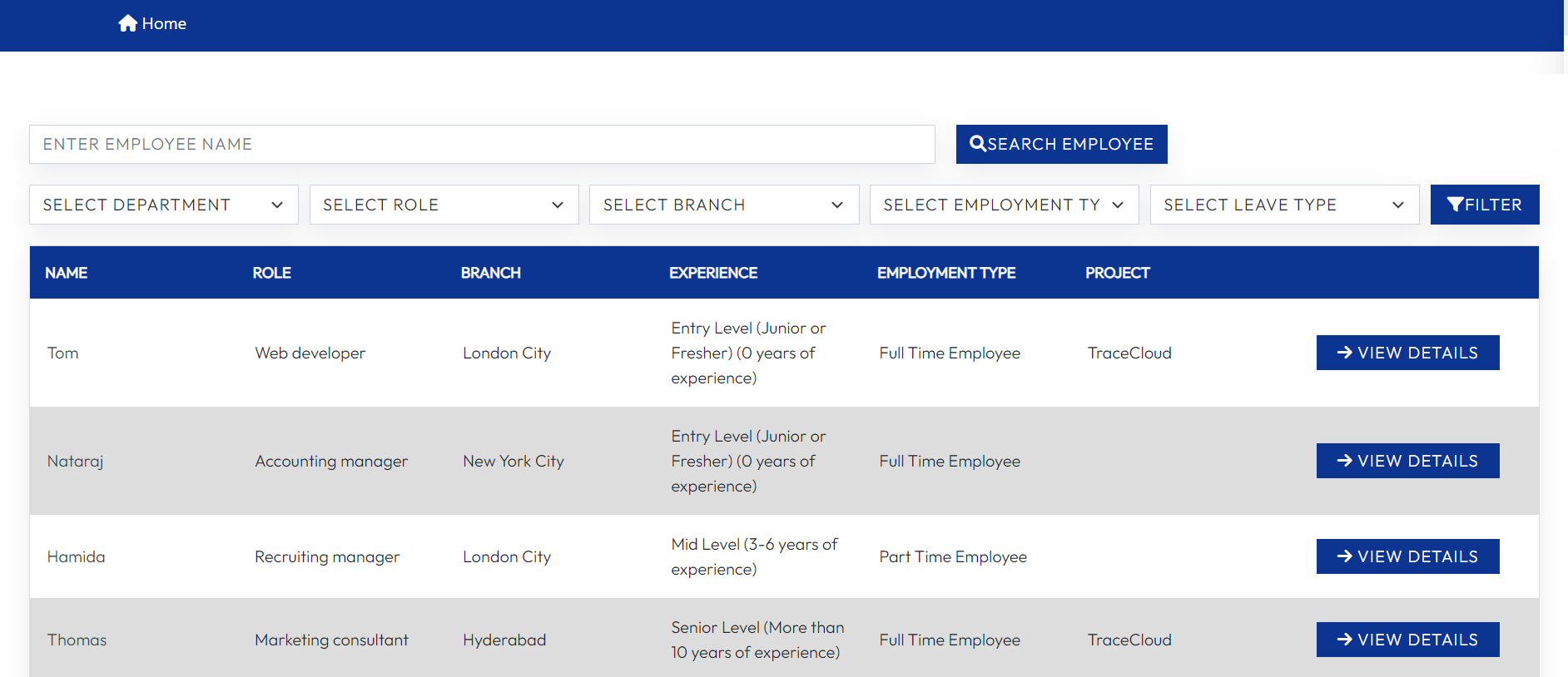


Figure Employee Management System - Homepage

The employee management system application consists of two main pages, i.e., the Homepage and Description page. The homepage in the application consists of a navbar, filter section, and table. The navbar consists of one menu option ‘home’. This option helps users to reload the current page or redirect to the homepage when the user is on the employee description page. The filter section on the homepage consists of one input text field known as basic search and five HTML select fields known as advance search. The basic search allows users to search for employees using their names. Once the user specifies the value in the input field and clicks the ‘Search Employee’ button, the ‘search\_results’ function in the ‘views.py’ file will get executed and returns the list of the employee. This basic search gives users the ability to search for the employee even if they don’t know the employee’s full name. The administrators or HR’s who are using this application can also advance search available on the homepage. This search will narrow the results and show the exact matches of the employee records. Once the administrator clicks the filter button, the ‘advance\_search’ in the ‘views.py’ will get executed and the options selected in the dropdown filters will be stored in a python dictionary and forwarded to the ‘get\_advance\_search’ function in the ’SPARQLMODEL’ class in the ‘models.py’ file. By default, the options in the advance search filters contain values specified as subclasses in the ontology design and the options available in the role dropdown contains all the roles available in the ontology. If the users select any department in the department filter, then the options in the role filter will get updated and show only the roles that are linked with the selected department. The view details button available in each employee record will redirect the user to the employee description page where the entire information about the employee is displayed.

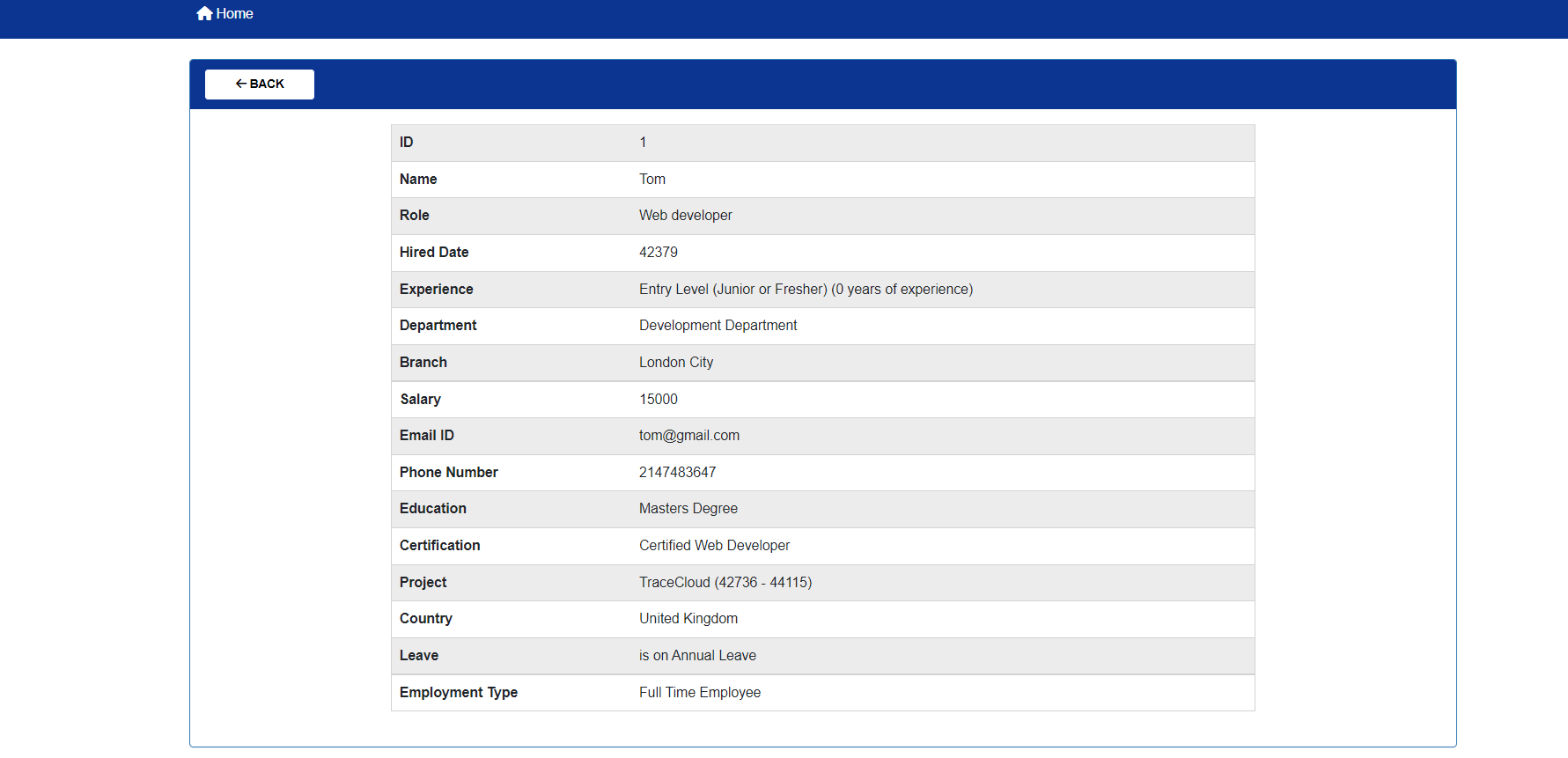


Figure Employee Management System - Description Page

The description page in the employee management system consists of a navbar and table. The ‘Home’ option in the navbar helps users to redirect to the homepage. The table on the description page shows lists of all object properties and data properties linked with the selected employee. This page will help the administrator or the HR to get an insight about the employee like projects that the employee currently working on, the certifications that the employee completed, the branch that the employee currently working in, and a few basic details like email-id, phone number, and country that employee resides in.

# Evaluation and Use

This section in this report will cover a list of SPARQL queries implemented in the employee management system. Few queries used in the functions such as ‘advance\_search’ are built dynamically when the user utilizes the features like advance search.

## Basic Search Query



Figure Basic Search Filter Query

Using the SPARQL endpoint, the ‘search\_employees’ function connects with the fuseki server using the SPARQLWRAPPER library. And the SPARQL query passed to the setQuery function will filter for employees whose name contains the value passed as the argument from the ‘search\_results’ view function. The BIND and STRAFTER functions will bind the values and return the substring, respectively. Below is the list of employees returned by the SPARQL query when the user searches for employees containing ‘th’ in the ‘name’ data property.

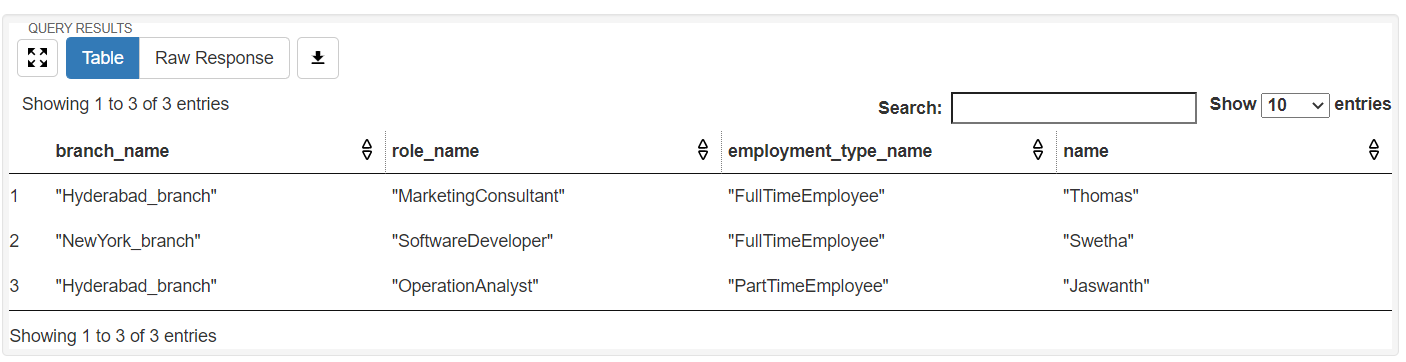


Figure Basic Search Query Results

## Advance Search Query

query\_string = """PREFIX ems: <http://www.semanticweb.org/44734/ontologies/2021/10/EMS#>  
 SELECT   
 (STRAFTER(?branch\_e, "http://www.semanticweb.org/44734/ontologies/2021/10/EMS#") AS ?branch\_name)  
 (STRAFTER(?role\_e, "http://www.semanticweb.org/44734/ontologies/2021/10/EMS#") AS ?role\_name)  
 (STRAFTER(?employment\_type\_e, "http://www.semanticweb.org/44734/ontologies/2021/10/EMS#")   
 AS ?employment\_type\_name)  
 ?name WHERE{  
 ?ID a ems:Employees;  
 ems:name ?name;  
 ems:works\_as ?role;  
 ems:works\_in ?branch;  
 ems:is\_a ?employment\_type.  
 ?department ems:has\_role ?role.  
 """  
if "branch" in filters:  
 branch = filters['branch'].replace(' ', '\_')  
 query\_string = query\_string + "?branch a ems:" + branch + ".\n"  
if "department" in filters:  
 dept = filters['department'].replace(" ", "\_")  
 query\_string = query\_string + " ?department a ems:" + dept + ".\n"  
if "role" in filters:  
 role = filters['role'].replace(" ", "")  
 query\_string = query\_string + "FILTER (?role = ems:" + role + ")"  
if "employment\_type" in filters:  
 type = filters['employment\_type'].replace(" ", "-")  
 query\_string = query\_string + "?employment\_type a ems:" + type + ".\n"  
if "leave" in filters:  
 leave = filters['leave']  
 query\_string = query\_string + "?leave a ems:" + leave + ".\n"  
if "name" in filters:  
 name\_con = "FILTER CONTAINS(lcase(str(?name)), '" + filters['name'].lower() + "')"  
 query\_string = query\_string + name\_con  
query\_string += """BIND (STR(?branch) AS ?branch\_e)  
 BIND (STR(?role) AS ?role\_e)  
 BIND (STR(?employment\_type) AS ?employment\_type\_e)  
}"""  
sparql = SPARQLWrapper('http://localhost:3030/EMS/sparql')  
sparql.setQuery(query\_string)  
sparql.setReturnFormat(JSON)  
results = sparql.query().convert()  
employees = []  
for result in results["results"]["bindings"]:  
 employees.append(result)  
return employees

The SPARQL query used in the ‘get\_advance\_search’ function is built automatically based on the filters selected by the users. For example, if the user selects the option from the employment type dropdown, then the conditions implemented in the function will automatically add the filter condition into the SPARQL query and returns the results that match with the filter condition.

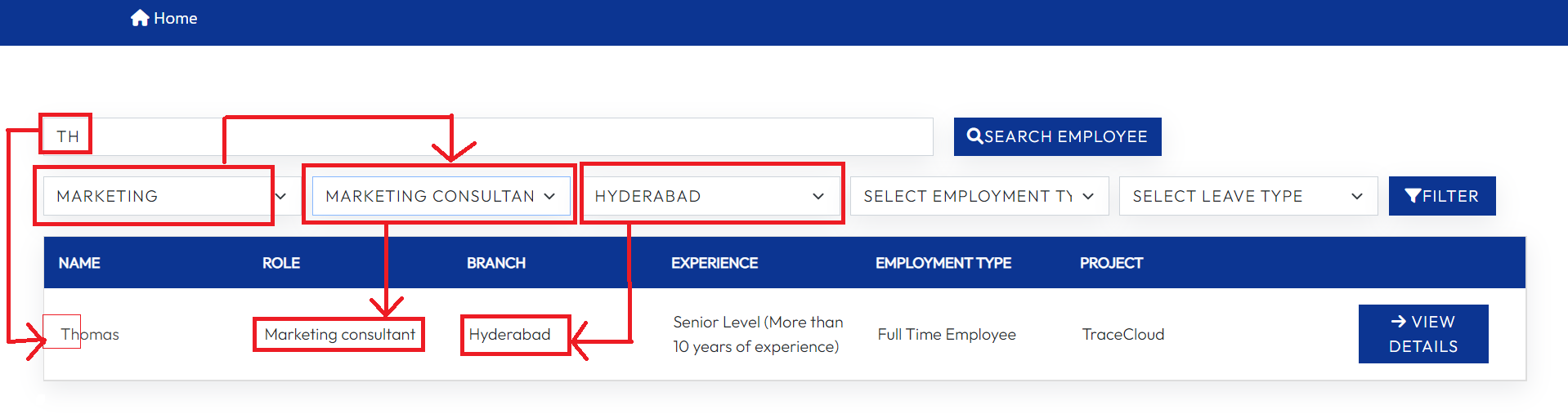


Figure Advance Filter Search Results

For example, in figure 5, first, the user queried for employees with ‘th’ in their names. And using the advance filter feature, the user narrowed down the results by selecting the ‘Marketing’ department in the departments dropdown and the ‘Hyderabad’ option in the branch dropdown. Once the users submit the request using the filter button, the system shows the list of employees that matched with selected options in the advance filters.

## Get Departments Query

The query implemented in the ‘get\_departments’ function will return all the subclasses listed in the department class. A few of the department subclasses defined in the employee ontology are ‘Human Resources’, ‘Development’, ‘Operations’, and more. The query is shown in figure 6 returns a list of all department's subclasses.

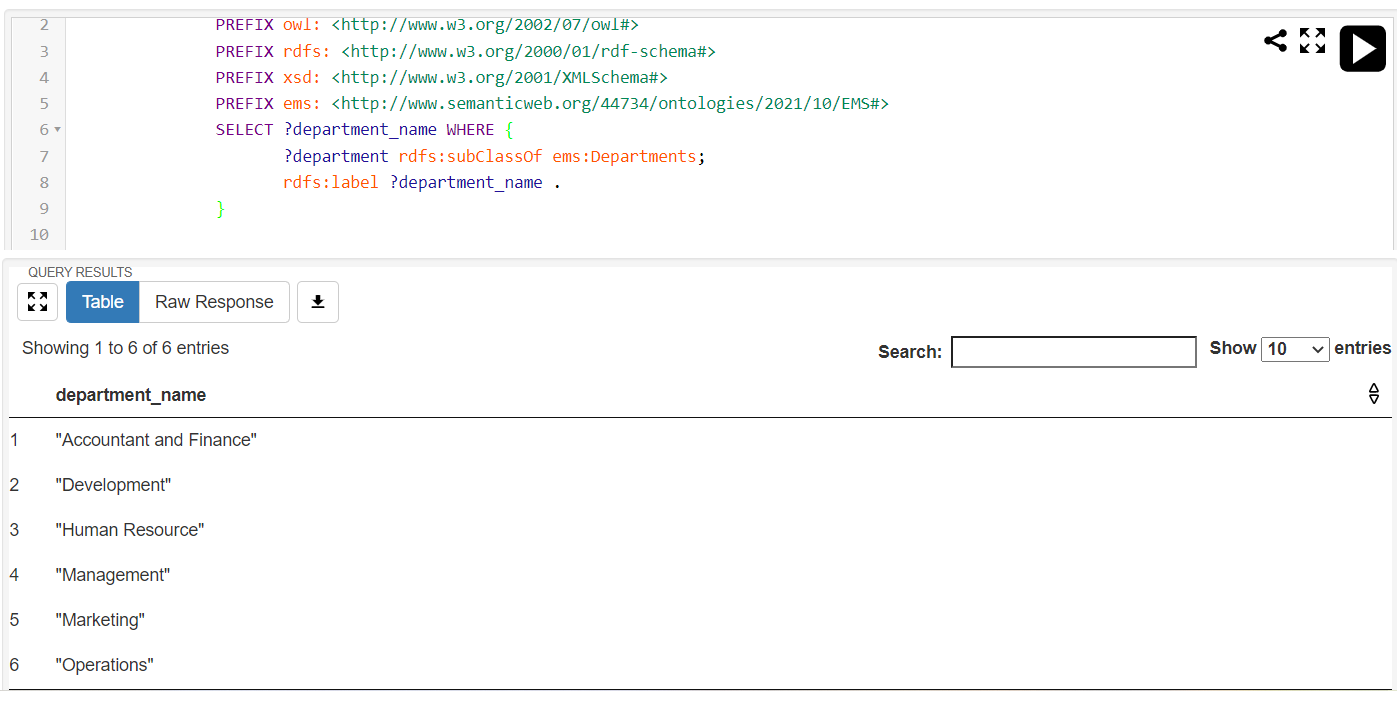


Figure Get Departments Query and Results

## Get Employment Type Query

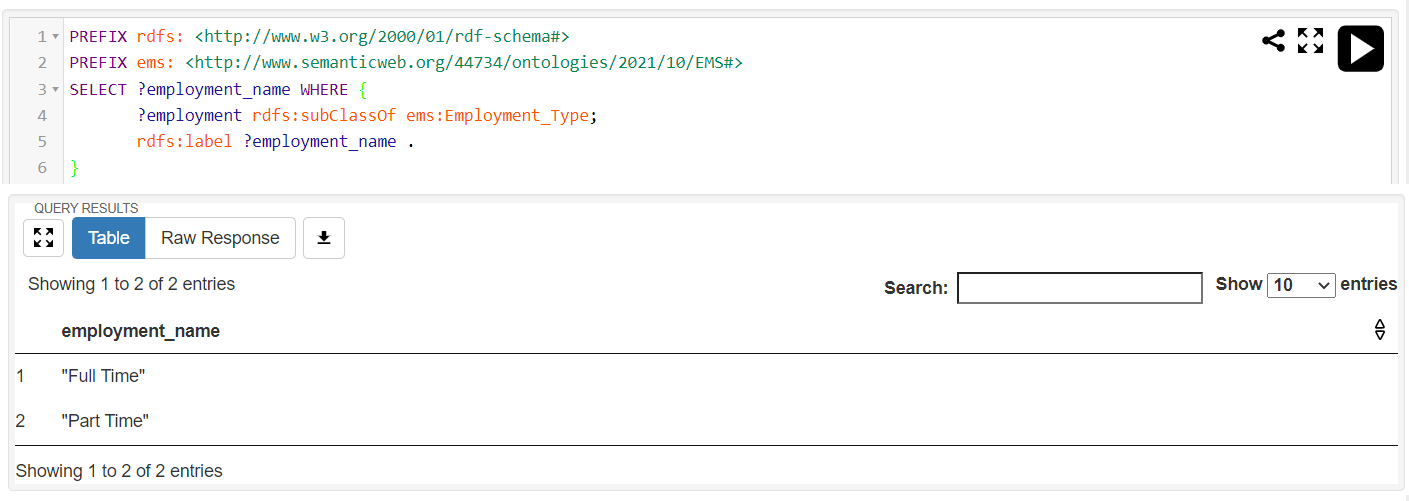


Figure 15 Get Employment Type Query and Results

The employment class in the ontology used in the employee management system is sub-categorized into two subclasses, i.e., full-time, and part-time. The query implemented in figure 7 retrieves all the subclasses from the ‘employment\_type’ class.

## Get Department Roles Query

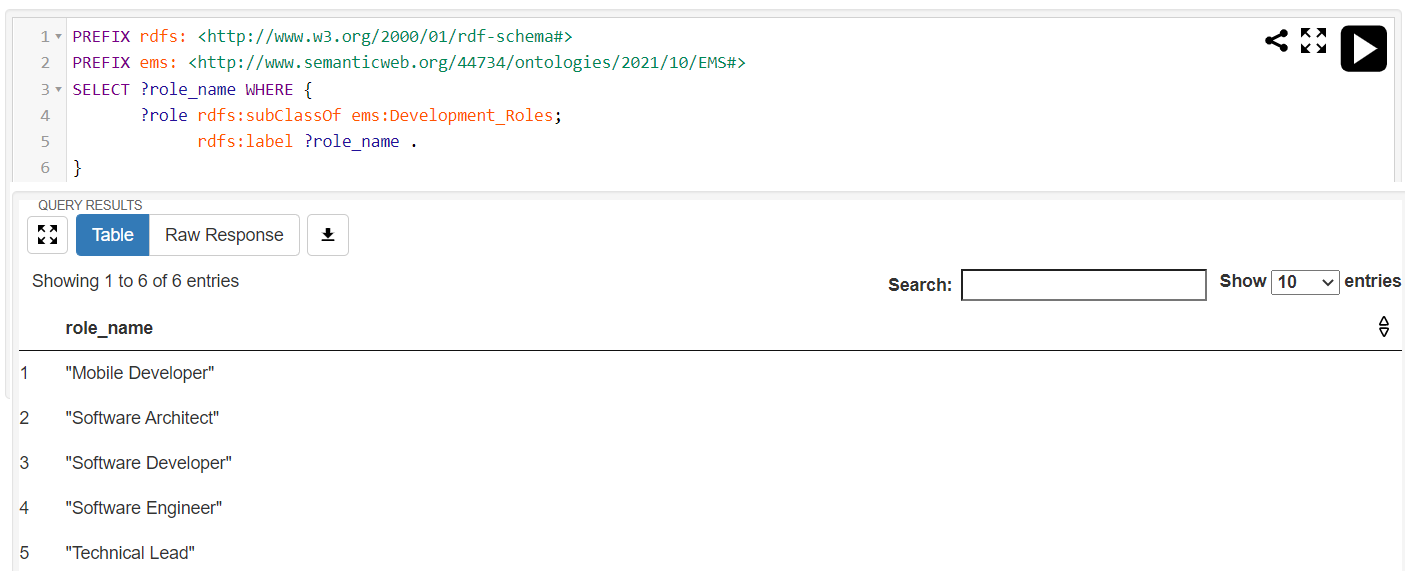


Figure 16 Get Department Roles Query and Results

The query implemented in the ‘get\_department\_roles’ function gets the roles related to the specific department. For example, the query written in figure 8 lists all the roles that are linked with the Development\_Roles class.

## Get Leaves Query

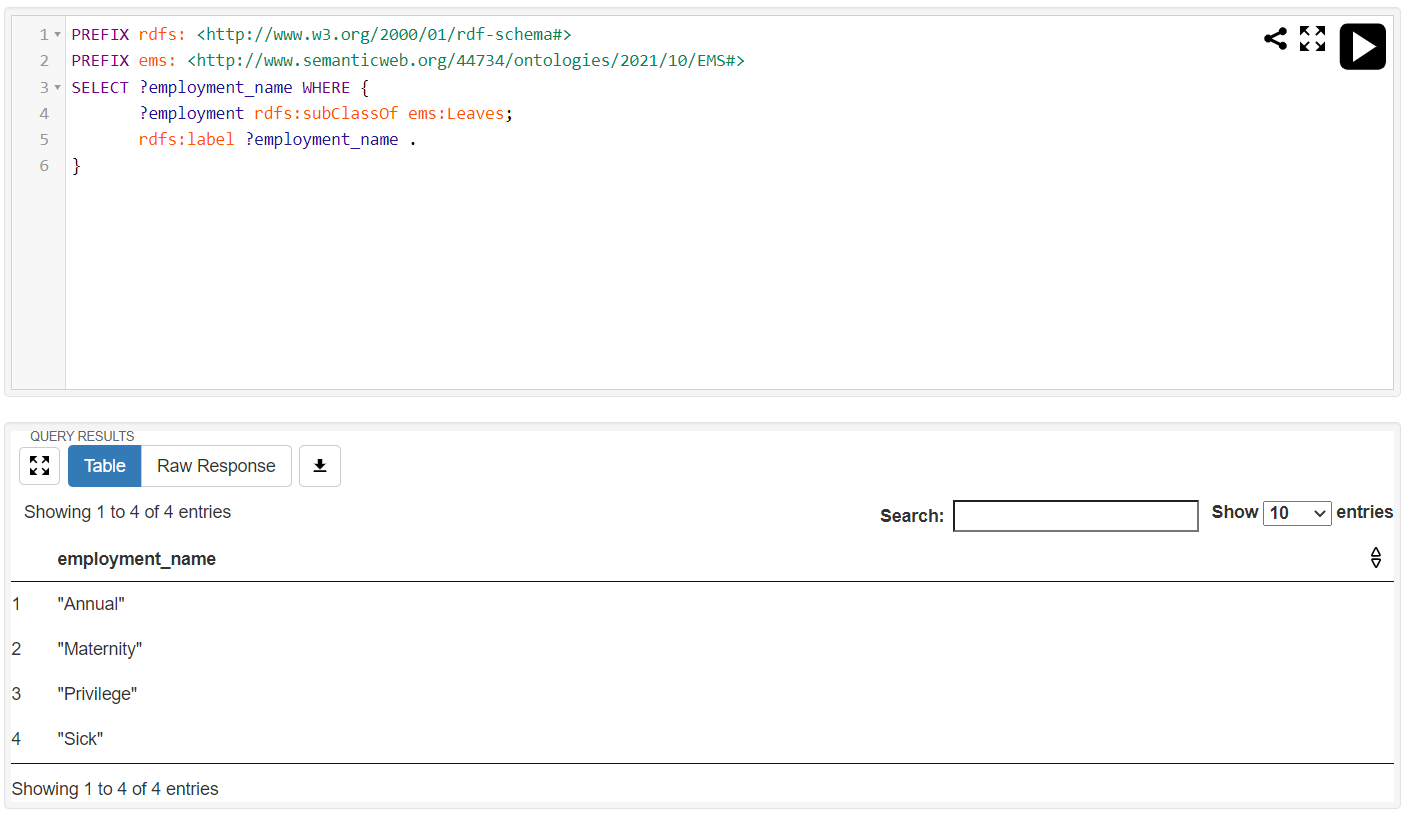


Figure Get Leaves Query and Results

The main concept of the leaves class defined in the ontology is to determine what type of leave the employee is in. And the query implemented in the get\_leaves function will retrieve all the types of leaves specified in the ontology.

## Get Employees Query

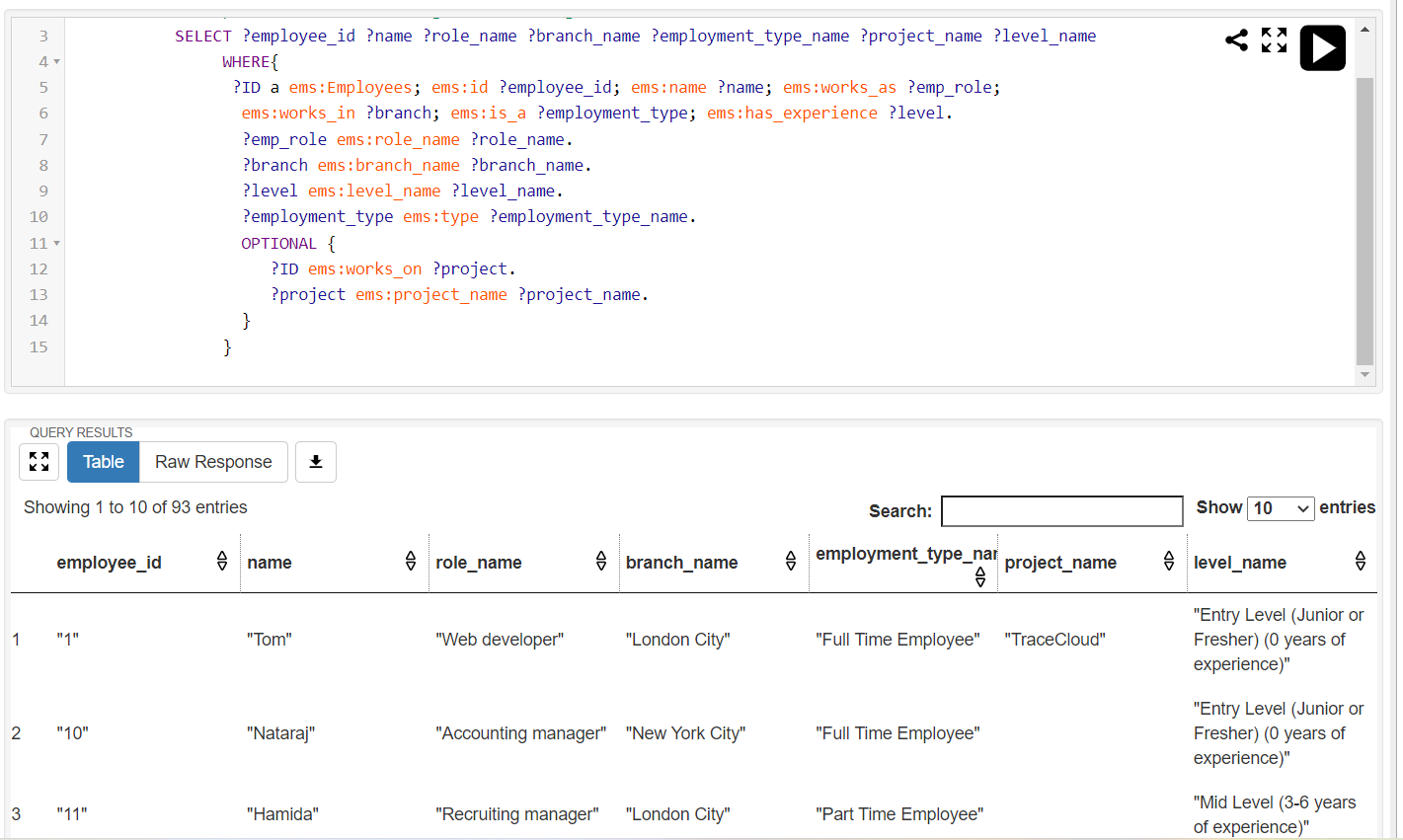


Figure Get Employees Query and Results

The query used in the ‘get\_employees’ function gets all employees' data that is available in the knowledge base. This query retrieves information like employee\_id, name, role, branch, department, and projects that currently working on. The OPTIONAL keyword used in the query returns the employee data even if the employee is not working on any project.

## Get Branches Query

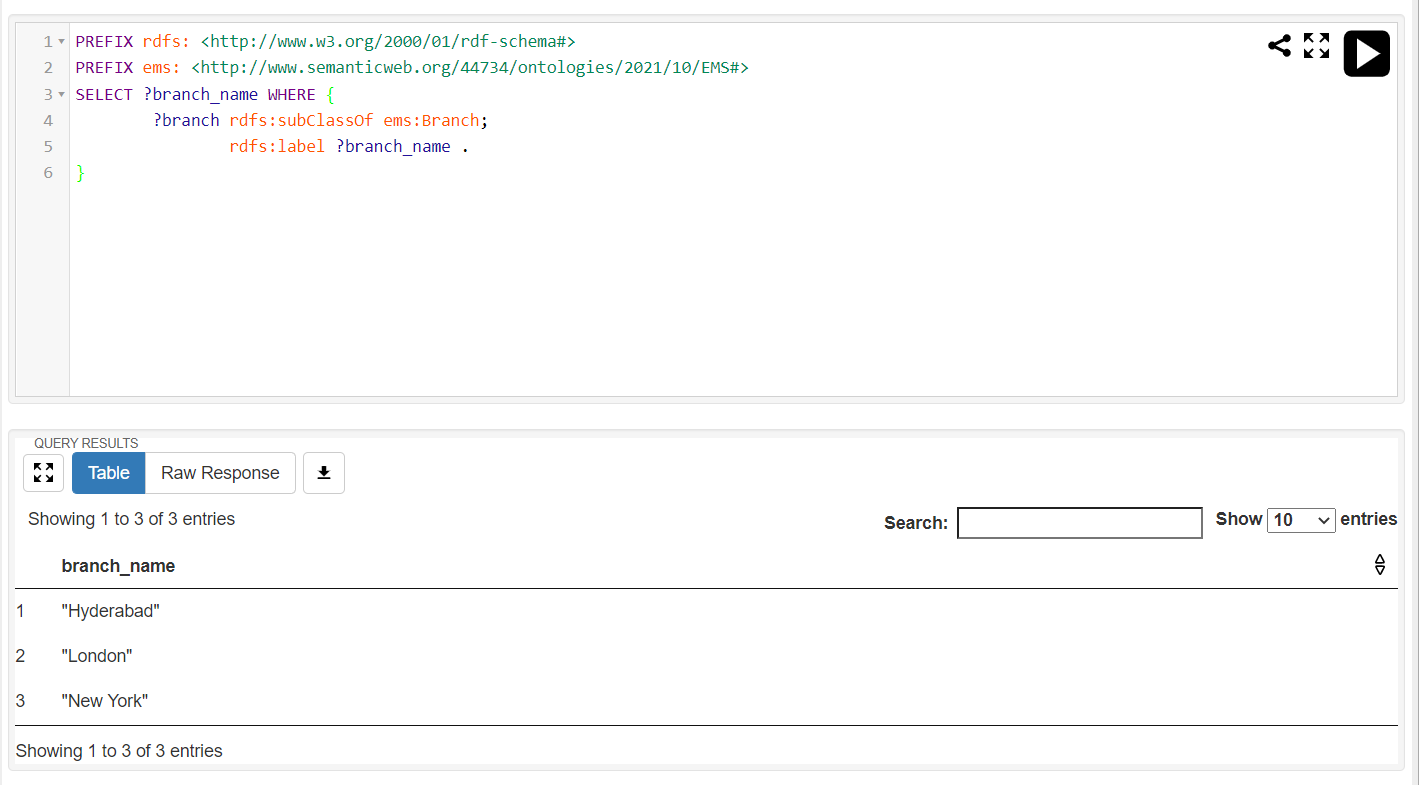


Figure Get Branches Query and Results

The functionality of the get\_branches function is similar to get\_departments and get\_employees functions. In this function, the SPARQL query returns all branches that are linked to the main class named ‘Branches’ in the ontology. The branch data retrieved from the server will later help the user to filter the employees based on the branch the employees are working in.

Before getting the results from the fuseki-server, the setReturnFormat function available in the SPARQLWRAPPER library allows the application to select the format of the data and convert it into the specified format using the convert() function. Once the data is converted into the JSON format, the required data is returned to the functions available in the ‘views.py’ file.

# Critical Reflection

Many companies in today’s era are using knowledge management systems. These systems will help organizations or businesses store and retrieve information in an organized way and help to achieve tasks by utilizing available knowledge. Currently, there are many types of knowledge base systems available on the internet including systems like organization or department-wide knowledge management platforms, learning management systems, customer service knowledge bases, and online community forums (Bloomfire, 2021).

Nowadays, many IT and non-IT companies are utilizing employee management systems to keep track of employees and the tasks that employees are involved in. These systems will help the companies to store the personal as well as professional details of the list of employees. Currently, there are many employee management systems with advanced features are available on the internet. A few of them are connecteam, Remote, SAP successFactors, Workday, and Saba, etc. (People Managing People, 2020). Most of these applications do not use the ontology model to retrieve information from the database.

The search feature used in current employee management systems or human resources systems retrieves information only by word matching instead of focusing on the meaning of the word (Jarrar, Vervenne, et al., 2007). Thus, results in getting irrelevant information. The employee management system proposed in this report uses semantic web technologies to search for employees by focusing on the meaning instead of word matching.

In the future, this ontology could be improved by implementing more object properties, data properties, and classes like skills and shifts, etc. This application currently focused on a few important aspects of the employee management system. Using this application, users can retrieve information using a few filter options like name search fields and other dropdown filters. This application can further develop with features like an advanced search field (able to search based on roles and department), a dropdown with multiple value selection that can retrieve information using different criteria and adding new employees or deleting existing employees from the system. Currently, the users can add or delete employees using Protégé.

# Conclusion

The report covers the aim, design, and implementation of the employee management system using semantic web technologies. This report gives information about the software, technologies, and programming languages used in this application. The ontology design section gives detailed information about the classes, object properties, and data properties implemented in the employee ontology. The implementation section covered details about the technology stack used like fuseki server, application structure like files and functions in each file, and user interface including features like search and dropdown filters available for use. The evaluation and use section provides a list of SPARQL queries used in the application. Each query SPARQL query is explained using code snippets and results obtained by running those queries on fuseki-server. The critical reflection section covers further developments of the application as well as existing employee management systems that are currently available online.

# Reference

Jena.apache.org. 2021. Apache Jena - Apache Jena Fuseki. [online] Available at: <https://jena.apache.org/documentation/fuseki2/> [Accessed 03 December 2021].

JetBrains. 2021. PyCharm. [online] Available at: <https://www.jetbrains.com/pycharm/download/> [Accessed 03 December 2021].

Google.com. (2021). Google Chrome Web Browser. [online] Available at: <https://www.google.com/intl/en_uk/chrome/> [Accessed 03 December 2021].

Microsoft.com. 2021. Microsoft Excel Spreadsheet Software | Microsoft 365. [online] Available at: <https://www.microsoft.com/en-us/microsoft-365/excel> [Accessed 03 December 2021].

Naz, K., Ali, Q. and Siddiqui, I., 2019. A Comprehensive Ontology for Human Resource Management in the Context of Software Industry. [online] ResearchGate. Available at: <https://www.researchgate.net/publication/334559640_A_Comprehensive_Ontology_for_Human_Resource_Management_in_the_Context_of_Software_Industry> [Accessed 03 December 2021].

Protege.stanford.edu. 2021. protégé. [online] Available at: <https://protege.stanford.edu/products.php> [Accessed 04 December 2021].

Chandra, R. and Guntupalli, C. (n.d.). 2008:PR002 User Interface Design -Methods and Qualities of a Good User Interface Design. [online] Available at: <https://www.diva-portal.org/smash/get/diva2:215020/FULLTEXT01.pdf> [Accessed 04 December 2021].

Bloomfire. (2021). Examples of Knowledge Management Systems. [online] Available at: <https://bloomfire.com/blog/knowledge-management-systems/> [Accessed 06 December 2021].

Features, P.H.I.R. the usage, tips, the best optimization of our software through blogs O. articles also incorporate the tricks, Employees, S.T.W.H.B.I. the P. of T. and Growth, T.O. (2020). Importance, Benefits, & Types Of Employee Management System. [online] Empmonitor Blog. Available at: <https://empmonitor.com/blog/employee-management-system/> [Accessed 06 December Dec. 2021].

People Managing People. (2020). 10 Best Employee Management Systems In 2020. [online] Available at: <https://peoplemanagingpeople.com/tools/employee-management-system/> [Accessed 06 December Dec. 2021].

Jarrar, M., Vervenne, L., Maynard, D. and P. Schmidt, A., 2007. The Semantic challenges and opportunities in the Human Resources domain. [online] https://www.researchgate.net/publication/242742864\_The\_Semantic\_challenges\_and\_opportunities\_in\_the\_Human\_Resources\_domain. Available at: <https://www.researchgate.net/publication/242742864_The_Semantic_challenges_and_opportunities_in_the_Human_Resources_domain> [Accessed 08 December 2021].