Keywords and Spatial Based Indexing for Searching the Things on Web

Muhammad R. Faheem[1*], Tayyaba Anees[1], and Muzammil Hussain[1]
[1]School of Systems and Technology, University of Management and Technology, Lahore, Pakistan
[e-mail: f2017288004@umt.edu.pk, tayyaba.anees@umt.edu.pk, muzammil.hussain@umt.edu.pk]
*Corresponding author: Muhammad Rehan Faheem

Received May 24, 2021; revised August 31, 2021; revised February 24, 2022; accepted April 30, 2022; published May 31, 2022

Abstract

The number of interconnected real-world devices such as sensors, actuators, and physical devices has increased with the advancement of technology. Due to this advancement, users face difficulties searching for the location of these devices, and the central issue is the findability of Things. In the WoT environment, keyword-based and geospatial searching approaches are used to locate these devices anywhere and on the web interface. A few static methods of indexing and ranking are discussed in the literature, but they are not suitable for finding devices dynamically. The authors have proposed a mechanism for dynamic and efficient searching of the devices in this paper. Indexing and ranking approaches can improve dynamic searching in different ways. The present paper has focused on indexing for improving dynamic searching and has indexed the Things Description in Solr. This paper presents the Things Description according to the model of W3C JSON-LD along with the open-access APIs. Search efficiency can be analyzed with query response timings, and the accuracy of response timings is critical for search results. Therefore, in this paper, the authors have evaluated their approach by analyzing the search query response timings and the accuracy of their search results. This study utilized different indexing approaches such as key-words-based, spatial, and hybrid. Results indicate that response time and accuracy are better with the hybrid approach than with keyword-based and spatial indexing approaches.

Keywords: API, Geo-spatial, Indexing, JSON, JSON-LD, Schema, sensor Things, Solr, Things.

1. Introduction

The Internet turned the world into a global network where every person or machine is connected to the Internet, also known as the Internet of Things (IoT). The objective of IoT is to connect every single object in real life with the Internet. Previously, only computers were connected to the Internet, but the IoT focuses on connecting the Things to the Internet [1]. IoT facilitates the user to communicate with the Physical Things to use the resources and services provided by the Physical Things. The W3C, Web of Things (WoT), enables interoperability across IoT Platforms and application domains. Overall, the goal of WoT is to preserve and complement existing IoT standards and solutions [2].

Several issues are faced in IoT and WoT, such as a lack of open standards, searching, interoperability, security, and scalability. The problem discussed in this paper is searching for devices. As WoT is based on IoT and devices, they continue to join and leave the IoT environment according to the user's location; thus, searching becomes complex. Previously, searching was mainly static in WoT [3], and now dynamic searching is emerging. The problem with static searching is that results are not always accurate because the information on which the search is based is outdated. The results are not always the best concerning user location and needs. Search results may be inaccurate due to outdated or stale data.

The current study has focused on solving the search problem by focusing on dynamic searching of devices in WoT to return the most relevant results and improve the accuracy of search results. By the accuracy of search results, the authors mean the most beneficial results for the user. For instance, if a user is searching for a coffee restaurant, results should return nearby locations of coffee restaurants, and results should not miss a newly opened restaurant, which may be the case with static searching.

In this paper, the authors have integrated "smart" Things into the Internet according to the Web of Things (WoT) architecture [2]. Considering "smart" things, a Thing has a description. Thing Description describes the metadata and interfaces of Things [2]. A Thing is an abstraction of a physical or virtual entity that provides interactions and participates in the Web of Things. According to W3C, Thing Descriptions provide a set of interactions that are based on a small vocabulary due to which integration of various devices becomes possible, and it allows diverse applications to interoperate. By default, Thing Descriptions are encoded in JSON format, allowing JSON-LD processing. The latter provides a robust foundation to represent knowledge about Things in a machine-understandable way.

The web has become an essential part of searching for anything by people using search engines, for example, AltaVista, Yahoo, and Google [4]. A search engine is a platform where people can access the information or data they want to access. The WoT architecture has different layers, including the Access, Find, Share, and Compose layers. This paper focuses on the FIND layer of the web of things architecture provided by Guinard [2]. The FIND layer consists of elements such as the REST crawler, Web Thing Model, Search Engine, Schema, JSON-LD, and RDFa. This paper focused on the search engine, crawling, and modeling of Things Description according to the model of W3C. Fig. 1 shows layer 2 (Find Layer) of the web of things architecture [2].

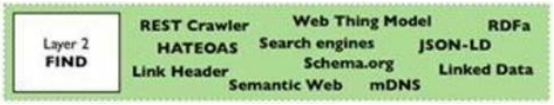

Fig. 1. Layer 2 of the Web of Things architecture Developed by [2].

To search the data from a search engine efficiently, one of the significant elements of a search engine is indexing [5]. Indexing is a place where search engines can store the data organized. So when the user query demands a search engine to find the data, instead of crawling individual pages, the search engine can find the data and return the result immediately to the user with the help of indexing. There are many tools available for indexing in which Things Description can be made, such as Locate32, Lucene, and Solr.

Locate32 is an open-source tool for finding Windows 98 and XP files. It finds the file by indexing it on the user's system hard drive. In Locate32, the user, first of all, creates a database to index the files. Once the database is created, Locate32 indexes the files to make the search more efficient and rapid. It is an indexing tool used to locate only local system files. It cannot work for web files or remote files.

Lucene is another tool available for indexing, an open-source tool for searching documents. It is a library based on java used for storing, retrieving, and searching data. In Lucene, many pre-build applications are available to structure the document. Lucene can index data of any type, written in any language based on the structure defined by the user. Lucence can index and search different HTML, MS Word, PDF, and XML documents.

Solr is an open-source tool [6] that is available for indexing the Things Description defined by the researchers. Solr is efficient compared to Lucene, as Solr can store and search a large amount of data, especially the data that continuously changes. Solr uses Extensible Markup Language (XML) and HyperText Transfer Protocol (HTTP) for configuration. The Solr works at the top of the Lucene.

Schema is how the researcher wants the data to be organized. If the Schema of things is not appropriately defined, then the indexing of Things Description is not efficiently performed. Therefore, Schema is very important in indexing the data. **Fig. 2** shows the example of modeling the Things Description according to the model of W3C [7].

```
{"@context": ["https://w3c.github.io/wot/w3c-wot-td-context.jsonld",
{"iot": "http://iotschema.org/"}],
"name": "LightControl",
"@type": ["Thing", "iot:LightControl"],
"base": "http://example.com:8080/mylamp",
"domain": ["iot:Building"],
"interaction": [
    {"name": "SwitchStatus",
    "@type": [
        "Property",
        "iot:SwitchStatus"
    ],
    "observable": false,
    "schema": "boolean",
    "writable": false,
    "form": [
        {"href": "/switch",
        "mediaType": "application/ld+json"}
    ]}
]}
```

Fig. 2. Schema for light control according to W3C Model [7].

Schema plays a vital role in searching the Things for any web of Things search engine. Without defining the Schema, the Description of the Things is not correctly indexed [8]. By

defining the Schema of Things, we can get only relevant values, making the search more efficient.

Things Description can be explained in different formats such as JSON, JSON-LD, and XML. JSON (JavaScript Object Notation) transfers and saves the data. Things Description with properties in JSON format is expressed as Sensor Properties, DataStream Properties, Location Properties, and Historical Properties. JSON-LD (JavaScript Object Notation) is used for linked data. Things Description with properties in JSON-LD format is expressed as ThingInformation, sensorType, and sensorURL. Things Description is expressed with properties in XML (Extensible Markup Language) format as name, identifier, Quantity and Station Location.

Query plays a vital role in information retrieval [9]. An efficient query results in the effective retrieval of information, due to which the response time is also optimal. Query building is the primary area representing the quality of search engines [10]. In most search engines, the query relies on the "NOT," "OR," and "AND" operators [11]. **Table 1** represents the usage of these operators in popular search engines like Google, Yahoo, and Live search.

Table 1. Boolean Operators Usage in Search Engine

Search Engine	Operator Used	Default Operator
Live Search	NOT, OR, AND	AND
Yahoo	NOT, OR, AND	AND
Google	OR, NOT	AND
Ask	OR, NOT	AND

Nowadays, the entire search engines are working on the Boolean AND operator. In the case of keywords and Description, our search engine works on the OR Boolean operator. Our search engine works on the AND Boolean operator in the case of name, type, location, and coordinates. For example, if a user searches the sensor whose location is Antarctica, the search engine returns the result of sensors located in Antarctica.

The Discovery of resources is a step toward finding the Things. To find the Things, there is a need to describe the resources as a description and then discover the resources using some protocols like HTTP. Currently, the resources are discovered in two ways. One way is to connect the resources by using some applications directly. In this case, the resources are discovered within a limited range. This process is a tremendous challenge in searching the physical Things because physical Things require the resources to be available publicly. At least there should be some mechanism available that can make the resources available at less cost [12-13, 14].

The second way is to get the resources using the Machine 2 Machine communication mechanism. In Machine 2 Machine communication mechanism, the machine can request the resources from some other machine or ask for some information from other machines. For instance, if a user wants to search for a coffee machine on WoT search engine, the search engine displays a list of coffee machines. The user can select any coffee machine and start communicating with that machine to know the resources, such as the quantity of coffee made by a machine. If coffee is not available, the machine can communicate with other machines to know their resources. If the other machine has available resources, the first machine recommends communicating with the second machine with the coffee available. This process is a machine-to-machine communication mechanism. However, there are a lot of problems with a machine to machine communication that can occur while getting the resources information from Machine 2 Machine, such as prolonged response time, scalability, interoperability, and heterogeneity.

Web of Things search engines is of different types, including a keyword-based search engine, a spatial-based search engine, and a hybrid approach. This paper utilizes hybrid approach.

A keyword-based search engine is the search engine that performs the searching based on some keywords such as id, Description, name of Thing. The spatial-based search engine is the search engine that finds Things based on latitude and longitude. A spatial-based search engine has coordinates, sensors, sensor gateways, and applications.

Coordinate is the centralized point for accessing and exchanging the data between the sensors and applications. The sensor is the physical Thing used to access or sense the data. Sensor gateways are the elements through which data of the sensor is collected. Application is the graphical user interface, from where data of sensor is accessed and integrated with the application itself. Before going into the detail of this paper, here we discuss some differences between Traditional search engines and WoT search engines.

In **Table 2**, similarities and differences in traditional and WoT search engines are described in different search engines. It can be seen that even the popular search engines such as Google, Yahoo, and Ask are doing static searching, which can miss the search results of dynamically added new devices.

Table 2. Similarities and differences in Search Engines

Similarities/Differences	Traditional Search Engine (Google, Yahoo, Ask etc.)	Web of Things Search Engine
Similarities		
A huge amount of Alternative web documents of same keywords available.	✓	✓
Index the data based on keywords.	✓	✓
No of Documents are increased day by day.	✓	✓
It provides free as well as paid documents.	✓	✓
Differences		
Only fixed devices	✓	
Fixed and Mobility devices		✓
Indexing on only keywords	✓	
Keywords and Spatial Indexing		✓
Crawls individual documents	✓	
Crawls indexing place instead of physical devices		✓
Depends on software	✓	
Depends on software and hardware		✓
Ranking on quality of contents and number of inbound and outbound links	✓	
Ranking based on functionalities and services		✓

2. Motivation

The principal objective of this research is to improve the dynamic searching of devices in WoT environment. Several parameters can improve searching, such as searching by keywords only, by name, type, URI, or using multiple parameters together. The paper also analyzes searches based on keywords, spatial, and hybrid methods. Over the past few years, many researchers have been working on this objective and identified several challenges such as dynamicity, diversity, and storing the changed data. These challenges motivated authors to solve the challenge of the dynamic searching of WoT devices.

3. Problem statement

With the changes in technology, emerging environments such as IoT and WoT, and the ever-growing data becoming big data, it has become challenging to search data, and searching the Things is a new research problem. Searching Things by using simple HTML pages or traditional methods used by search engines is not appropriate for searching the "smart" Things in WoT on a country level, province level, and city level. Even it is impossible to search a Thing worldwide with a simple query. For efficient searching, indexing and ranking are very important. Without indexing and ranking, search engines cannot search the Things accurately and promptly in a WoT environment which is the research problem discussed in this paper. For this study, experiments were done, and the findings are discussed in the results of Research Methodology section. The results of previous studies were not empirically available; therefore, we cannot compare current findings with the previous works.

4. State of the Art

In the last few years, many researchers have worked on the FIND layer of the WoT architecture [3-15, 16, 17]. They have created the Things Description, and some of them have proposed the indexing of Things descriptions to improve searching. This paper categorized the working of previous studies as follows:

4.1 Keywords based Indexing

Hoang-Vu [18] proposed the Schema for indexing the Things Description based on the keywords. His proposed indexing schema handles only name and description-based queries. His proposed indexing structure cannot handle the spatial-keyword queries.

Khodaei [19] proposed an index structure based on the keywords. His proposed index structure considers the time factor but does not consider the spatial factor. Nepomnyachi [20] developed an index structure based on keywords and spatial, but it does not work for the devices when their location is changed.

Ostermaier [21] developed a search engine named "Dyser," which indexes based on keywords. His proposed search engine does not work on dynamic searching of devices.
Wang [22] developed a search engine named "Snoogle." Snoogle first registers the devices and then saves the devices' data in the database. The drawback of Snoogle is that when physical devices change their locations, then Snoogle does not detect that change.

The authors in a past paper [21] proposed a search engine that performs indexing based on the Description of keywords. If a user enters the type of Thing, then the proposed search engine cannot find the Thing as search is only based on the Description of Things, and other

attributes are not considered.

Another study [23] performed the indexing based on the services performed by the Things. The problem in their proposed approach is that if a user searches the Things against the name, id, URI, then their proposed search engine cannot find the Things.

The authors in a previous research [24] developed a search engine that searches Things against the popular keywords saved in a database—for instance, air quality, temperature, weather, sensor, etc. The drawback of their indexing technique is that data is not efficiently retrieved. For example, in a research [23], if the user searches the Thing by name, the search engine cannot find the Thing even if the Thing is in the database. The problem of non-efficient retrieval of data occurs.

Nadeem [23] developed a system based on keywords and services based on indexing. His proposed system indexes the descriptions of the Things according to the keywords and services, but his system only works for static devices.

Authors in a prior study [12] discovered the resources by registering the devices and resources. By registering the resources with the system, the authors then discover the resources by using matching descriptions. Another research [25] discovered the resources according to the users' requirements. However, this technique is not suitable for the dynamic nature of devices because many resources are not discovered by using this technique.

Another paper [13] discovered the resources by providing a mechanism to register the sensors with the registry option, which is centralized. This technique also uses RFID tags, Bluetooth, and QR codes, for registering the resources, but this does not work correctly for heterogeneous devices.

4.2 Spatial based Indexing

Li [26] proposed an IR tree based on spatial indexing. His proposed index tree retrieves the documents based on the keywords and the spatial keywords, but his proposed index tree does not consider the time factor.

A past study [27] performed spatial indexing based on latitude and longitude. If a Thing changes its location, then his proposed technique does not find the Thing, and the proposed search engine cannot find the Thing against the complete location of the Thing. The authors in another study [25] searched based on the temporary location of a Thing and the combination of the Description of the Thing. However, if the Thing changes location, the user cannot find the data. The other problem with his proposed search engine is that if the user enters the address of the Thing, then the proposed search engine cannot find the data.

4.3 Hybrid Approach (Both Keywords based Indexing and Spatial Based Indexing)

Butler [28] proposed a system that included both keywords and spatial indexing, but his proposed system can work for only fixed location devices. Different authors have worked on keywords and spatial-based search engines. A previous paper [29] proposed storing the data of Things in a Web-enabled store. Their proposed architecture work for both keywords-based as well as spatial-based searching. The author used the WoT scripting API for performing operations and actions on a Thing.

Another research [30] proposed the hypermedia multi-agent system to search the user's query. Their proposed architecture used the SPARQL query to search against the keywords and spatial searching. The drawback is the maintainability and availability of the agent system. Another study [31] performed hybrid searching by using Node.js. However, the issue with

their proposed architecture is that the schema of Things Description is not adequately maintained. Due to this, the system does not support the data.

Another research [32] presented a spatial-based search engine. The sensor owner uploads the sensors' data using a web-based graphical user interface in the proposed search engine. The architecture proposed by authors [32] for search engines is shown in **Fig. 3**. The architecture consists of coordinates, sensors, sensor gateways, mobile proxy, and applications. The user interface was a significant issue raised in the proposed search engine [32]. The output of searching was shown in the form of maps and nodes. Such interfaces were not user-friendly, thus it became complicated for non-technical persons to understand them. The other challenges and issues faced in the search engine were the heterogeneity and scalability factors. As the number of sensors increased, it became difficult to get the sensor data. In addition to this, it also became challenging to index the data.

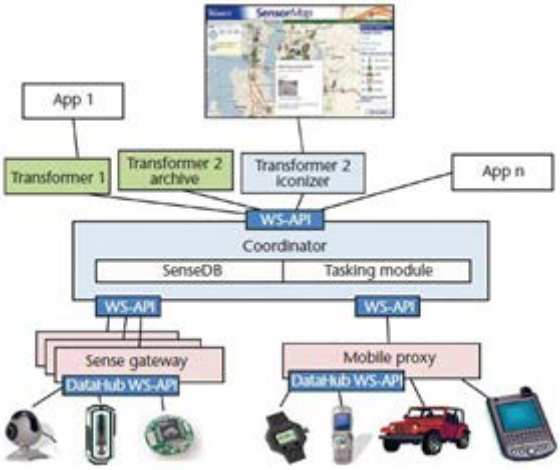

Fig. 3. Architecture of Grosky [32]

Authors in [33] developed a "MAX" system for facilitating humans to search the physical Things based on the keywords. The keywords in the proposed search engine were the tags. The proposed architecture of the author [33] is shown in **Fig. 4**. In their proposed system, the user can search the tag-based Things and locate them in the location of the user's choice. Tags were marked as public or private. Any user can search the things based on public tags, but the private-based tags were only searched and accessed by the owner of Things. The benefit of the MAX system is that the output is shown in the user-friendly interface, but the drawback of the MAX system is that locating the Things is dependent on users' choice, which does not result in efficient searching. As the Things are indexed according to the user's choice, there is no proper mechanism to index the Things, which is another drawback.

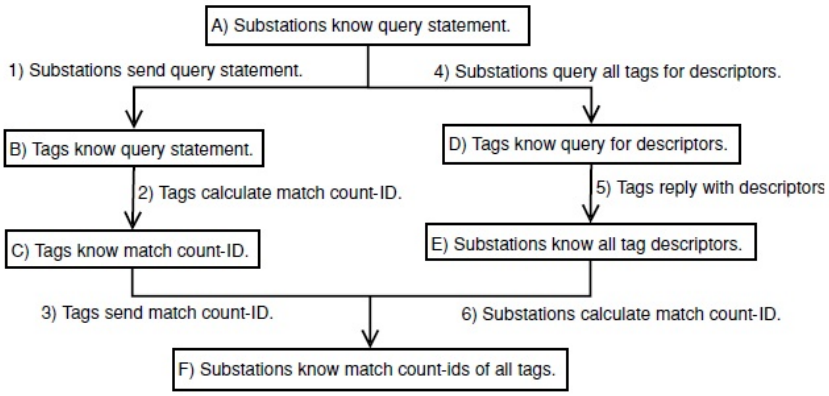

Fig. 4. Architecture of Yap [33]

In this paper, the authors have used the JSON, XML, and JSON-LD Schema to describe the resources and use the HTTP protocol to discover the resources. The method discussed in this paper is meant to search the devices and Things and covers the dynamic Discovery of Things and machines.

In WoT, some authors have used the static methods of indexing and ranking for searching Things in WoT. However, the dynamic nature of devices is not considered. Previously searching the Things in WoT, there was a problem related to data availability. Data was not available due to poor Schema. The authors in this paper have created their Schema according to the description model of W3C JSON-LD and the sensor Things' open-access APIs. In this paper, the authors have developed a search engine for dynamic searching of the location of the Things in WoT based on Keywords and Spatial Indexing. **Table 3** shows the work of different authors on indexing.

Table 3. Working of Authors on Indexing

Author Name	Indexing	Working of Author	Drawbacks
Aswale [34]	Keywords based indexing	A survey on the issues in WoT search engines	Dynamic searching in WoT is missing
Hoang-Vu [18]		Proposed a system focused on efficient response timings	Spatial queries not handled
Khodaei [19]		Proposed a system that indexes only keywords.	Spatial queries not handled
Nguyen khoi Tran [35]		Proposed a system for searching the Things through web interface	Location of the mobile devices is not updated
Ostermaier [21]		Proposed a search engine based on Keywords	Dynamic searching in WoT is missing
Sciullo [36]		Proposed a discovery application for WoT	GUI is not friendly
Shemshadi [37]		Proposed ThingSeek Search engine	Manual extraction of resources of Things
Shemshadi [34]		Proposed a Keyword and location based search engine	Manual update of the location of devices
Wang [22]		Proposed a search engine for Web of Things named as “Snoogle”	Dynamic searching in WoT is missing
Yuchao Zhou [27]		A survey on Searching Techniques	Dynamic searching in WoT is missing
Li [26]		Developed an IR tree based on keywords and spatial indexing	Dynamic searching in WoT is missing
Nadim [23]		Searching based on keywords and services provided by the Things	Searching is missing for mobile devices.

Nepomnyachi [20]	Keywords and Spatial based indexing	Proposed an index structure for searching the Things	Dynamic searching in WoT is missing
Butler [28]		Search engine based on the geographic properties of Things	Searching Fixed/static location devices only

Table 4 represents the work of researchers on keyword-based search engines and spatial-based search engines.

Table 4. Keyword based and Spatial based Search Engine

Name of Author	Keyword Based Search Engine	Spatial Based Search Engine	IoT Search Engine	WoT Search Engine	Working of Author
Wang [22]	Yes	No	Yes	No	Proposed a search engine named as Snoogle. In their proposed search engine, entity is described as a set of keywords. It cannot work for dynamic nature of devices as well as for the static nature of devices in large scale network.
Yap [33]	Yes	No	Yes	No	Proposed a system named as MAX. The MAX system uses tags to sense the entities instead of the sensors. MAX requires high communication speed as it has to broadcast the communication to each substation and each tag.
Ostermaier [21]	Yes	No	Yes	No	Proposed a search engine named as Dyser. In this search engine indexing is the main problem. But indexing works on the basis of only keywords and cannot retrieve the dynamic nature of devices.
Ding [38]	Yes	Yes	Yes	No	Proposed a search engine named as IoT-SVK. IoT-SVK performs the searching based on the spatial as well as on the keywords. However, it cannot pull the data of dynamic nature of devices.
Grosky [32]	No	Yes	Yes	No	Proposed a spatial based search engine, in which data of sensors is uploaded by the owner of sensor. The drawback of proposed system is the deficiency of user friendly interface and the problem of scalability.
Proposed Search Engine	Yes	Yes	Yes	Yes	Proposed a search engine which works for the dynamic nature of devices, as the proposed search engine indexes the devices on the basis of coordinates. It also maintains the history of location of devices in history location tag. The search engine does efficient indexing. Efficient indexing is done based on the schema defined for the things according to the schema model of the W3C JSON-LD. And all the above search engines are the Internet of Things search engines and the proposed search engine is the Web of Things search engine. All the above discussed search engine as well as the techniques discussed in table 1 are used for static searching. We developed a search engine that performs dynamic searching based on the relative location as well as the coordinates. By taking the both properties, the things are efficiently indexed and as results the output of searching is better. This work is not implemented in the papers discussed above.

5. Research Methodology

With the advancement in technology and IoT and WoT, physical objects can now connect to the Internet and provide their services on the web. Communication has enhanced between multiple devices, and more information can be accessed. The difficulty with this approach is that searching mechanisms previously used for searching were static**.** They used static data**,** which was primarily obsolete, resulting in wrong search results and missing the most relevant ones. The concept of dynamic searching has emerged because search results accuracy can be improved with it.

In this paper, the authors have proposed an indexing mechanism based on keywords and spatial indexing for better dynamic searching in WoT environment that is the need of the time as most of the applications are nowadays using IoT and WoT. We have used different parameters to analyze search results, including the type of Things, name of Things, Description of Things, location of Things, and geospatial coordinates.

We have implemented a searching algorithm for indexing Things in WoT and analyzed the system using different parameters with keyword, spatial and hybrid indexing approaches. We have also analyzed response timings and the accuracy of search results with all mentioned approaches. We have tested the system with different parameters such as several sensors, using name only, using multiple keywords, relative location, coordinates, and hybrid approaches to find optimal results. There is no empirical data available in WoT, so we have compared our system with different indexing approaches such as by keywords only, by doing spatial indexing only, and using a hybrid approach.

We have proposed a user-friendly interface for finding Things. The proposed search engine performs the inserting and the fetching of data to process the query entered by the user.

In WoT, each node, whether a sensor, actuator, or any physical device, is assigned a unique identification, known as a URL (Unique Resource Locator). On the web, all URLs should begin and be controlled by HTTP. We have used URI, Relative location, Name, and Coordinates as a URL. We have used HTML tags to link the complete address to make it a meaningful URL.

For connectivity of Things with the Solr interface, we have used the COAP protocol. We have used PHP to develop an indexing mechanism for a WoT search engine that dynamically finds the Things in WoT. **Fig. 5** represents the architecture of the proposed Web of Things Search Engine (WoTSE).

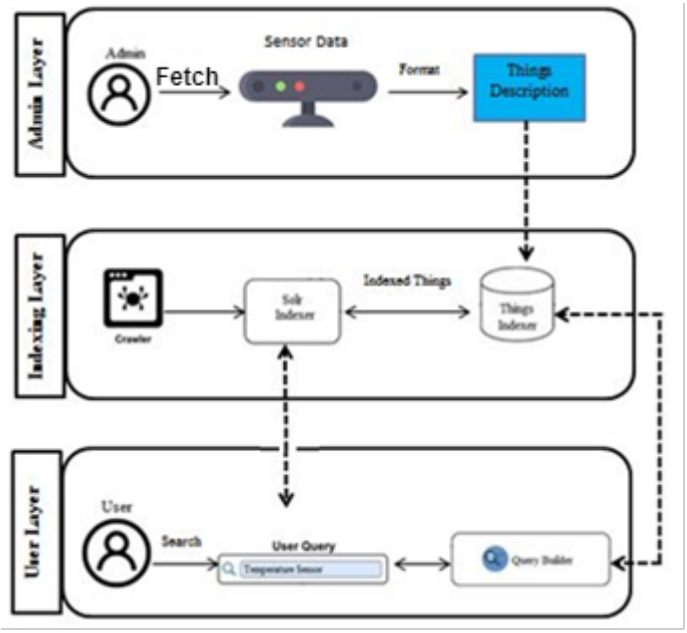

Fig. 5. Architecture of WoTSE

The algorithm of the proposed model is presented below with an explanation afterwards.

1. Schema→Schema defined in Solr
2. Da→Data of things
3. For→Format according to JSON-LD, XML, JSON
4. D→Description of things
5. If(Schema== D) then I→Indexed data
   Else
   Do→Data is not indexed
   End If
6. Q→Query of user
7. If(I==Q) then
   Di→Display the results.
   End If
8. Exit.

In step 1, we define the schema in Solr that indicates what type of data we will store and manage in Solr. Defining a schema in Solr is necessary. The data cannot be stored, managed, and retrieved without defining schema in Solr. In step 2, the proposed search engine (PSE) gets the data of Things. The PSE gets the Energy Sensor data, Environment sensor data from Sensor Web APIs, Home sensors from W3C APIs, and Bicycle rental station Sensor APIs. The PSE retrieved the data from different sources, so the data format is also different, and it contains much-unneeded information. PSE format the data according to JSON-LD, JSON, and XML in step 3. Once we got the formatted data that we named Description of Things in step 4, we matched the description of things with the schema defined in Solr. In step 5, if data is matched, then Solr allows to store, manage and retrieve data. If it does not match, then it is not stored. In step 6, users enter the query in the query interface, and the query directs toward the Solr (step 7); if the query matches the data stored in Solr, the Solr displays the results.

The first layer is the admin layer, the second layer is the indexing layer, and the last layer is the user layer. The brief detail of these layers is as follows:

5.1 Admin Layer

The admin layer consists of two components: the first is the sensor data, and the second is Things Description. The admin performs the fetching and inserting of sensor data. The proposed model fecth sensors' data from the Sensor Web APIs and uses the "RESTFUL APIs." The RESTFUL API consists of HTTP and identifiers (URL). With the help of the HTML get function, the proposed model retrieves the current data of the sensor. With the help of the post function, the proposed model retrieves the changes in data.

Once the changes in data are retrieved, the proposed model uses the HTML put function to put the changes in data. For instance, consider the coffee machine that is connected to the sensor. The get function helps retrieve the current quantity of coffee in the machine or any other state. The post function is used to check if any change has occurred in the state or quantity of the coffee machine. If there is any change observed in the state of the coffee machine, then the data is updated again with the help of the put function. The data is fetched and then saved into the Things indexer by passing through the Things description. The fetched data is formatted into W3C JSON-LD [6]. Once the Things data are formatted, it is passed to the Things indexer, the first component of the indexing layer. The purpose of passing the sensor data to the Things indexer is to save sensor data.

5.2 Indexing layer

The proposed architecture consists of a user-friendly graphical interface, query builder, query parser, Things indexer, and Solr indexer.

User-Friendly Interface: In this architecture, the authors provided a user-friendly graphical user interface in which the user enters a query for searching the Things.

Query Builder: When a user enters the query, the query passes to the query builder. Then, the query builder processes it and passes it to the query parser.

Query Parser: The query parser parses the query according to the defined keywords and spatial-based query. Then, the query parser passes the query to the Things indexer.

Things Indexer: Things indexer matches the parsed query according to the Things Description. After matching, the Things indexer passes the query to the Solr indexer.

Solr Indexer: Solr indexer finds the data and returns the result to the query builder. Crawling of data is done at the Solr indexer. Query builder then displays the results to the user in a user-friendly interface.

The authors have defined the Schema according to Things' static and dynamic properties in this paper. The static properties include the Id, Description, name, Uri, type [14], and version, while the dynamic properties include coordinates and relative location. In coordinates, we have defined the location in latitude and longitude. In the relative location, we have defined the complete location of Things, including country, street, postal code, city name, [2]. **Fig. 6** shows the static and dynamic properties used in this paper.

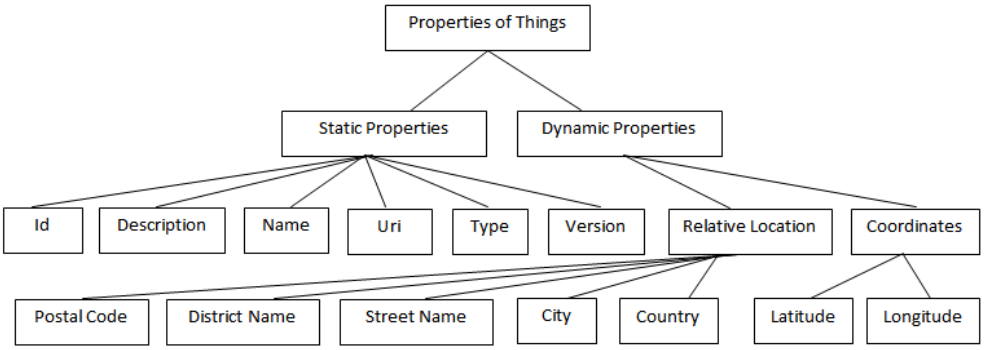

Fig. 6. Static and Dynamic Properties of Things

Once the Schema of Things is defined, the next step is to describe the Description in different formats. At this stage, we need to perform a parsing step. The parsing step is essential because the Description of Things can come in different JSON, JSON-LD, and XML formats. So, without parsing the Description of the Thing, the Things are not indexed in Solr. The authors have used the XML parser and the JSON-LD parser for parsing [39-40]. Some other formats describe the Description of the Things, such as DPWS metadata, micro formats, and WSDL. Search engines and even web browsers do not understand these formats, which are obsolete now. That is why we choose JSON, JSON-LD, and XML formats. Another reason for choosing these formats is that they are more understandable and describe a Thing more efficiently than the other formats, irrespective of whether it is a human or a device.

Once the Things are parsed, the next step is to index the Things according to the Schema. This step is performed to crawl the Things at the indexing place instead of crawling each Thing. Crawling each Thing is very difficult because physical things continuously change their locations. That is why the search engine performs the crawling at indexing to perform efficient searching.

If the data type of any element defined in the Schema of Things is not matched with the element in the Things description file uploaded by the authors, then the data is not indexed. In

this paper, the authors perform different types of indexing. The first one is Keyword-based indexing second is spatial-based indexing, and finally, the authors have used a hybrid approach for indexing. Keyword-based indexing is the technique used to index the Description of data based on some textual words.

For efficient data retrieval, the authors categorize the keywords into the name, URI, id, version, type, and Description (services) keywords to index the data. The benefit of categorizing the keywords is efficient searching. The user searches Things by using any of these keywords and combining more than one keyword. All the keyword-based indexing techniques discussed in State-of-the-Art Section shown in **Table 3** are based on one keyword. They do not use multiple keywords. Authors have used multiple keywords, due to which the search results are better than previous studies. The other indexing technique which is used in this paper is spatial-based indexing. The spatial-based indexing technique is the indexing technique that is used to index the Description of Things based on location.

To overcome the problems in spatial indexing, such as heterogeneity and scalability factors, the authors categorize location-based indexing into two types. The first is the relative location, and the other is the coordinates. The relative location consists of the complete location of the Thing, including street number, country, and postal code. Furthermore, the coordinates consist of Latitude and Longitude. Both the relative location and coordinates are used for the dynamic nature of devices. This paper indexes the Things based on both relative location and coordinates.

Authors have performed a search by using a hybrid approach and combining both of the approaches mentioned earlier, keyword-based and spatial and individually. The results are better using the hybrid approach than the other approaches, as shown in **Table 5**. When location changes, the user searches the Things and finds the new location using the historical data attribute. When the data is efficiently indexed, then the result of searching is also efficient.

In this paper, the authors have classified the query into three components one is a keyword-based query, the other one is a spatial-based query [14], and the last one is a hybrid query. The reason for classifying the query into components is to increase the efficiency of the search engines and to search the devices dynamically. The authors have built a query based on the type, name, Description, location, coordinates, and type of the Things along with the above discussed Boolean operators.

For dynamic searching, authors have saved the data of Things with the history of locations and the current location (latitude and longitude) at the place of indexing instead of the database for efficiency purposes. If authors save the data in a database, they need to crawl the actual site where the data is placed, which is not possible in the case of the dynamically changing devices. The authors have performed the crawling step at the place of indexing**,** as it gets the data when the location changes through a history-location tag.

In the fast-moving world, users want efficient searching mechanisms and the accuracy of search results. Considering these search requirements, the authors have analyzed queries' response timings and accuracy from three different perspectives. The authors have compared the results obtained using keyword-based indexing, spatial indexing, and a hybrid approach. The accuracy means that the proposed approach finds the most relevant data required by the user. The authors calculated the accuracy by using the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The output consists of positive and negative elements, dependent on TP, TN, FP, and FN.

1. Positive: The output is positive if the search engine searches the result according to one of the three perspectives, including keywords, spatial, and hybrid.

2. The output is negative if the search engine finds the result opposite to the perspective or if the search engine does not find the Thing. For instance, if a user searches the Thing by name and the name of Thing is not found, the search engine finds a different Thing. Then, in this case, the result is negative.

5.3 User layer

The user layer is responsible for searching the Things against the query entered by the user. The user layer consists of two components: the first is the User Interface, and the other is the query builder. The user interface is connected with the query builder. The user interface is implemented in PhP. When the user enters a query in the user interface, the query goes to the query builder. The query builder builds the query according to the name, location, type, Description, and coordinates of Thing and the "AND" or "OR" Boolean operator. In addition to this, the query builder processes the query based on three components: the Keyword-based query, the spatial-based query, and the hybrid query. After processing the query, the query builder sends the query to the Things indexer to get the actual data. The Things indexer gives the data to the query builder against the processed query. The query builder gives the data to the user interface to get the required output, which the user wants.

5.4 Experimental Setup

The authors have used the Solr tool for indexing a Thing. Solr is an open-source tool available for indexing Things against the Schema defined by the users. We have used the Solr tool and the Solarium plugin [14]. A Solarium is a plugin that is used for the connection of Solr with PHP.

5.5 Results

We have analyzed the response time and accuracy of searching results with keyword-based, spatial, and hybrid indexing approaches. We have uploaded the Description of 750 sensors in Windows 7, using the 32-bit operating system. After 750 requests system was not scalable anymore. The browser was unable to process requests, and requests were timed out. Results were taken concerning response timings and accuracy separately. We have performed different experiments to check the validity of our algorithms. For example, if a user searches the Things by 'Type query,' our search engine returns the expected accurate results. There is a need to check the accuracy of search engines because there can be faulty cases where accurate search results are not returned. For instance, there might be a case where the user demands a 'dock sensor' for searching, but the search engine results provide data of 'thermal temperature.' Another case might be that when the user demands 'proximity sensors', the search engine yields only ten sensors as search results while there are 200 proximity sensors available. It would be performing faulty searching. We performed different experiments to deal with such inaccurate cases and assess our algorithm's working. Results are discussed in the following subsections:

5.5.1 Response Time

Two elements are essential for any search engine: indexing and ranking. In this paper, we worked on indexing and created a search engine that searches the dynamic Things. If there is optimal indexing, then the searching will also be optimal. When searching Things for users, the most crucial concern is the accuracy of search results and response time. In this paper, we have written an algorithm (on page 13, under section 4, Heading 5.2 Indexing layer) to search the dynamic Things in WoT environment. To evaluate our search engine, we had compared our search results with literature, but as WoT is a new domain, we did not find empirical/statistical results. So, we analyzed our search engine by considering its accuracy and acceptable response time for end-users. We created replicates to find the optimal level (number of replicates) until we got the most accurate search results. Response time for search should be acceptable to the users,

so we also tested our system with replicates and analyzed response time. In WoT, information comes from different sensors. We intended to find out that if we use more sensors and more things are indexed, would it affect search results, such as accuracy being reduced or search time being increased? Further, we used different parameters to analyze how search results can be improved, for instance, by using things names, descriptions, etc.

Initially, only one Thing was indexed and increased up to 10 Things with an addition of only one Thing at each stage. No significant response time difference was noticed in that case. In the next step, 10 Things were indexed in a single phase with no significant difference in response time. The process continued with 20, 30, 40, and 50 Things indexed at the same phase, and a significant difference in response time was noticed in the case of 50 Things. At this level, we stopped further testing because we observed that response time had exceeded that will not be acceptable for users.

If our algorithm returns the search results in 5 seconds to a user against 'Name query' and the same algorithm yields search results of 'description query' in 2 seconds, it would be better to use the 'description' for searching the Things. The sole purpose of designing the replicates was to provide a search engine for users to display Things in a short period.

We performed keyword indexing based on single keywords as well as based on multiple keywords. In addition to this, we also performed spatial indexing based on relative location coordinates, and used a hybrid approach as well, which is a combination of both approaches. The authors divided the work into different cases, which are as follows:

Primary Case: Keyword Indexing

Sub-Case1: Name.

In case 1, the authors indexed the Things on the "name" keyword. **Fig. 7** shows the response time of indexing the Things based on the name keyword and the number of Things indexed.

Sub-Case2: URI

In case 2, the authors indexed Things on the "URI" keyword. **Fig. 7** shows the response time of indexing the Things based on URI, the keyword, and the number of Things indexed.

Sub-Case3: Description

In case 3, the authors indexed Things on the "description" keyword. **Fig. 7** shows the response time of indexing the Things based on the description keyword and the number of Things indexed.

Sub-Case4: Type

In case 4, the authors indexed Things on the "type" keyword. **Fig. 7** shows the response time of indexing the Things based on the type keyword and the number of Things indexed.

Sub-Case5: Hybrid Keywords (Uri, name, description, type)

In case 5, the authors indexed Things based on “URI,” “name,” “description,” and “type” keywords. **Fig. 7** shows the response time of indexing the Things based on combining URI, name, description, and type keywords and the number of Things indexed.

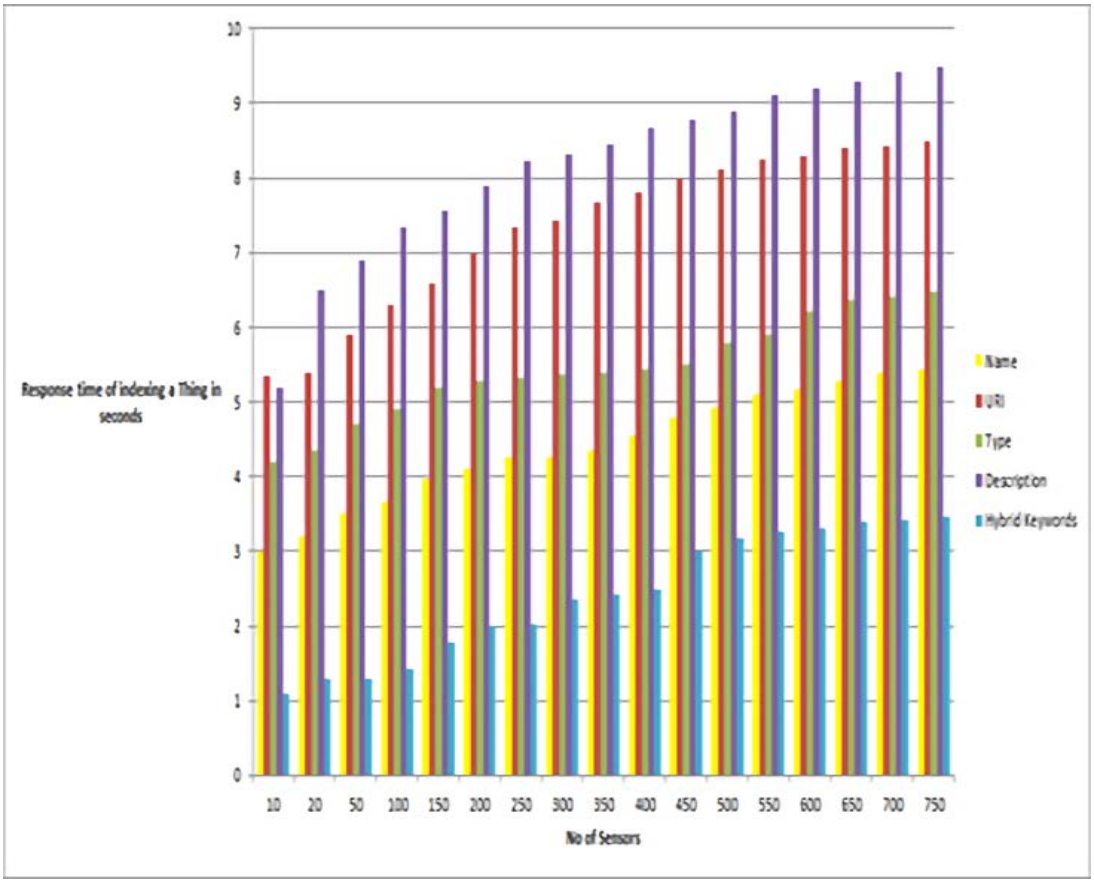

Fig. 7. Response time of indexing a Thing in seconds using Keyword

After analyzing response timings, the results indicated that the best approach was using hybrid keywords where response time was below 3.5 seconds. In contrast, the worst response time was with a Description where the response time exceeded 9 seconds which would not be acceptable for the users. In our opinion, a response time above 5 seconds is not generally suitable for a search query.

Our results indicated that response timings depend on the number of sensors. With 500 sensors, Name can be used for a query. The query with URI and Description did not appear to be suitable parameters for a search query because the response time was more than 5 seconds, even with ten sensors. Query with Type can be used for up to 100 sensors. Query with Hybrid approach looks good as the response time was below 3.5 seconds even with 750 sensors. We recommend using the hybrid approach for indexing in WoT for response timings based on our results.

Primary Case: Spatial indexing

Sub-Case1: Relative Location

In case 1, the authors indexed Things on “Relative Location” information. **Fig. 8** shows the response time of indexing the Things based on Relative location and the number of Things indexed.

Sub-Case2: Coordinates

In case 2, the authors indexed Things on “Latitude and Longitude” information. **Fig. 8** shows the response time of indexing the Things based on Coordinates and the number of Things indexed.

Sub-Case3: Combination of Both Relative Location and Coordinates.
In case 3, the author indexed Things by combining both Relative and Coordinates information. **Fig. 8** shows the response time of indexing the Things based on relative location and coordinates and the number of Things indexed.

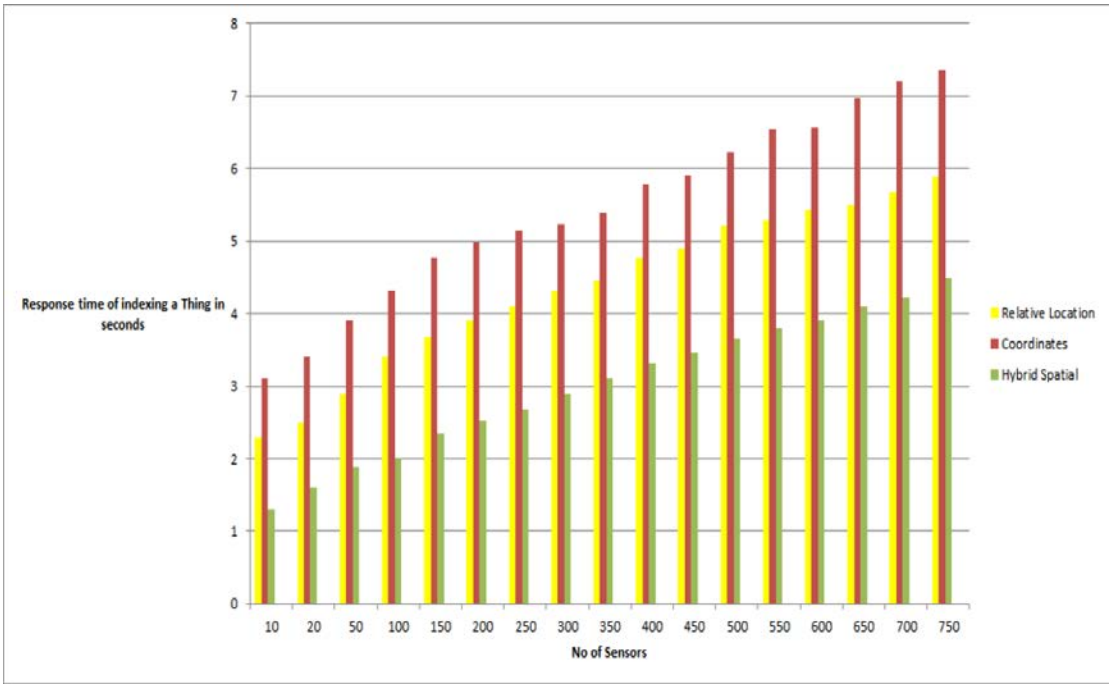

Fig. 8. Response time of indexing a Thing in seconds using Spatial

After analyzing the results, hybrid spatial seemed the best approach, which had a response time below 4.5 seconds and the worst approach was to index the "things" based on coordinates that had a response time above 7 seconds with 750 sensors which was above our ideal acceptable response time of 5 seconds. However, with up to 200 sensors, response time with coordinates should be acceptable to users. With a relative location of up to 450 sensors, response time stayed below 5 seconds. Using the Hybrid spatial indexing approach, response time stayed below 4.5 seconds; we recommend using the hybrid spatial indexing approach.

Response timings in all test cases were highest when searching was based on coordinates. Response timings with relative location were also higher when a hybrid approach was used, but they were better than coordinates. Results indicated a rise in response timings with all test cases when the number of sensors was increased with coordinates and relative location. The authors suggest using a hybrid approach with 100 sensors because, in this case, response time is within 2 seconds, which should be suitable for user experience.

Primary Case: Hybrid Approach for indexing
The hybrid approach of indexing a Thing combines case 5 of keyword indexing and case 3 of spatial indexing.

Sub-Case: Keywords and Spatial Indexing
We indexed using the hybrid keywords, hybrid spatial, and hybrid (combination of hybrid keywords and hybrid spatial) approaches. **Fig. 9** shows the response time of indexing the Things based on these approaches.

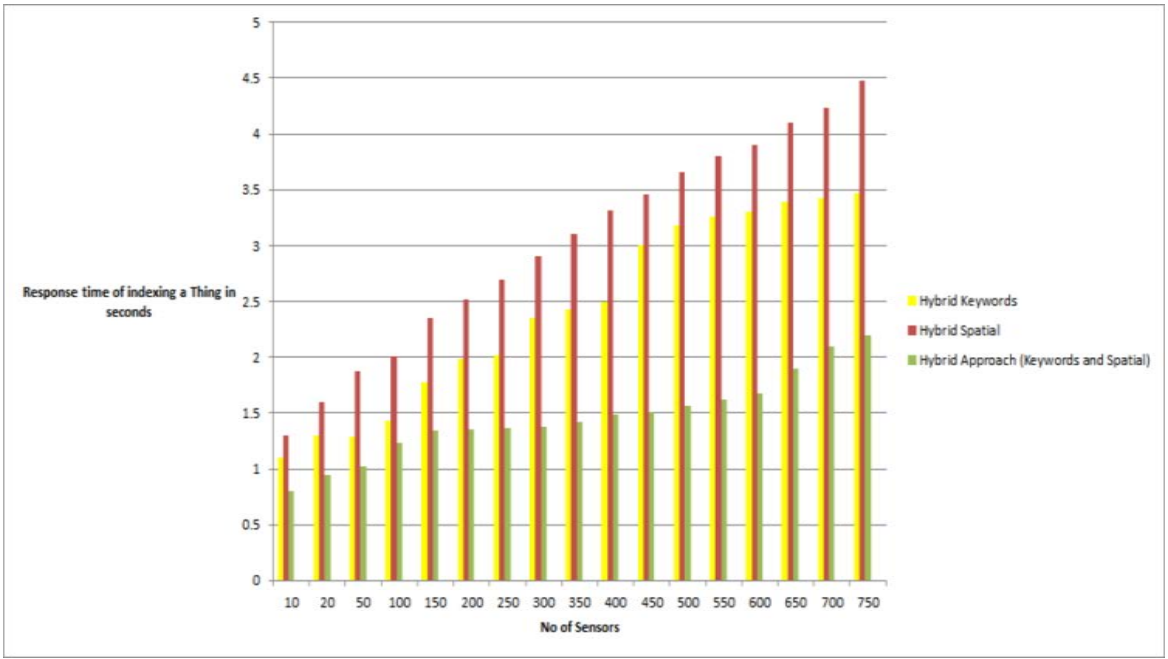

Fig. 9. Response time of indexing a Thing in seconds using Hybrid Approach

Our results indicated that response time stayed below 4.5 seconds with all the hybrid approaches using up to 750 sensors. However, the best results were obtained using the hybrid (hybrid keywords and hybrid spatial) approach. Response time stayed below 2.3 seconds even with 750 sensors, so we recommend using this approach for indexing. Using hybrid keywords also gave response timings below 3.5 seconds which is also good. Hybrid spatial indexing with 750 sensors gave the worst response timings, below 4.5 seconds, which would also be acceptable for users in general.

Moreover, in all the above-discussed cases and subcases, the response time slightly increases when the number of Things increases. The best approach is a hybrid approach with a response time below 3.5 seconds, and the worst approach is to index the "things" using the hybrid spatial approach, which has a response time above 4 seconds.

5.5.2 Accuracy

Authors have tested 20 different queries on 750 Things indexed in each case to check the accuracy of results returned by a search query. The authors have discussed some cases, and they have analyzed the accuracy of searched data against all the cases discussed below:

Primary Case: Keywords Searching

For checking the accuracy of results returned by a search query, different cases were used for keyword searching**,** including searching by name, URI, Description, and Type individually. **Fig. 10** shows the accuracy of results in keyword searching. Also, searching was performed by using Hybrid keywords. Seven hundred fifty sensors were used in all cases.

Our results indicated that using hybrid keywords accuracy was 67%. Results also indicated that accuracy was about 58% when we used URI. The results with names were similar; about 57 % accuracy was achieved. Searching the "Things" using the "Type" keyword had 48% accuracy, the lowest among all indexing cases. Our results indicated that searching the Things was better in the case of indexing multiple keywords in comparison to searching by using single keywords.

Sub-Case1: Name Searching
When a user searches for data, then much-unrelated data is shown. For instance, if a user wants to search for a temperature sensor or a contact temperature sensor, in this case, the search engine shows the results of all temperature sensors indexed in a system instead of showing only the contact temperature sensor.
Sub-Case2: URI Searching
When Things are indexed on URI, a user needs to remember the URI of Things. A user cannot remember the URI of a large number of Things. If a user cannot remember URI, he/she cannot search the data.
Sub-Case3: Description Searching
In the case of Description, there is a chance that many Things have the same Description. If multiple Things have the same Description, it is tough for a search engine to find accurate results.
Sub-Case4: Type Searching
The drawback of this case is that the user can search only relevant data Types. For instance, if the user wants to search a data of sensors subtype such as non-contact temperature sensor, the user cannot find the actual results because the Type keyword indexes the general type of sensors instead of their subtype. The available sensors include a temperature sensor, proximity sensor, gas sensor, dock sensor, etc. To search for a sub-type of a sensor, adding a sub-type to the indexed Description of Things is essential.
Sub-Case5: Hybrid Keywords (URI, Name, Description, Type Searching)
In this case, the user can search any Thing, any subtype of Thing, any description of a sensor, and any specific name of a Thing.

After analyzing the results, the best approach concerning the accuracy of results returned by a search query was using the hybrid keywords, which had 67% accuracy, and the worst approach was to search the "Things" by using the "type" keyword, which had 48% of accuracy.

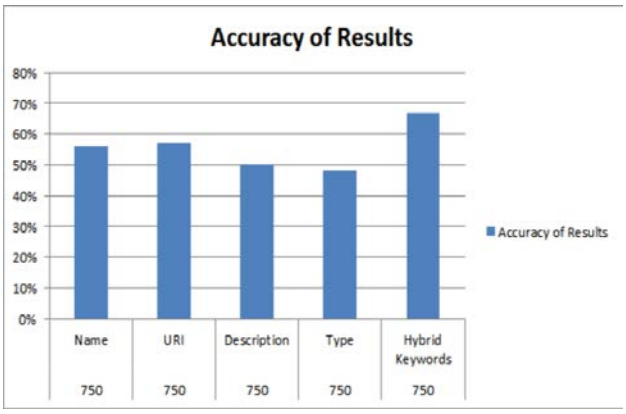

Fig. 10. Accuracy of Results using Keywords

Primary Case: Spatial Searching
Sub-Case1: Relative Searching
In this case, the user searches the Thing's city, country, and street. But all this information is static. If a sensor changes its location, then the user cannot search the exact location of the sensor. Relative information is updated with the help of Coordinates.
Sub-Case2: Coordinates Searching
In this case, if the sensor changes its location, the user quickly finds the sensor with the help of coordinates. But in this case, the user cannot find the sensor's city, country, and street location.
Sub-Case3: Hybrid Spatial (Relative Location and Coordinates Searching)

In case3, we have indexed the sensors based on both "Relative Location" and "Coordinates." By combining them, the user can find the sensors with the updated information of its relative location and the coordinates.

Fig. 11 shows the accuracy of results in Spatial Searching. After analyzing the results, the best approach was hybrid spatial, which had 62% accuracy and the worst approach was to search the "things" by using the "relative location," which had 47% accuracy. With coordinates, accuracy was 51%, making it better than the relative location.

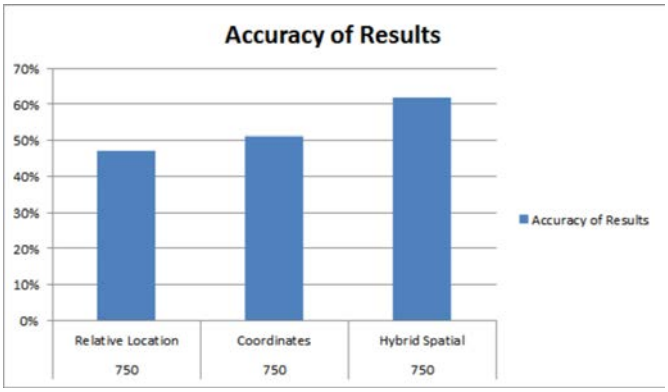

Fig. 11. Accuracy of Results using Spatial Approach

Primary Case: Hybrid Approach
Sub-Case: Keyword and Spatial Searching
Using a Hybrid approach of indexing, the results of searching are improved compared to all other approaches. For instance, if a user searches data by entering some keywords and a sensor changes its location. Then with the help of spatial information, the search engine quickly finds the sensor. In this case, the user will still get the desired result. **Fig. 12** shows the accuracy of results in Hybrid Searching.

Our results showed that with the Keywords based approach, accuracy was 68%. We obtained the best results with a hybrid approach with an accuracy of 79%. We obtained the worst results for searching the "Things" using the hybrid spatial approach with 62% accuracy. We recommend using the hybrid approach for indexing based on our results.
Results for accuracy with TP, TN, FP, and FN are shown in **Table 5**.
Below is the description of how the authors calculate the Accuracy in each case:

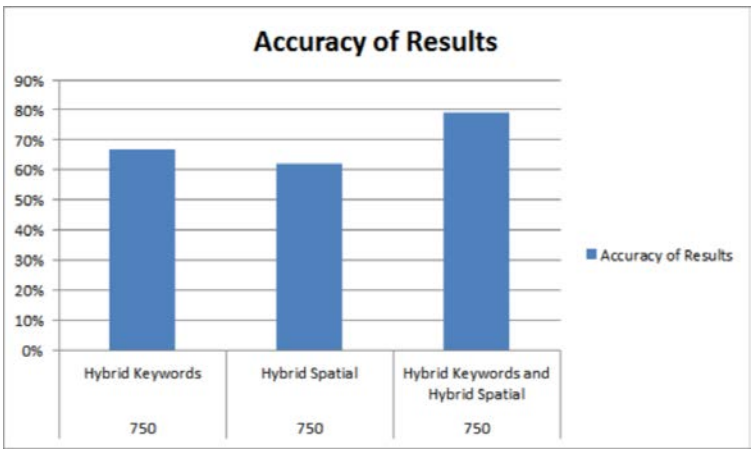

Fig. 12. Accuracy of Results using Hybrid Approach

Table 5. Results for accuracy with TP, TN, FP, and FN

		Positive		Negative		Accuracy
		TP	FP	FN	TN	
Cases	Name	93	416	27	214	56%
	URI	420	62	169	99	57%
	Type	307	184	173	86	48%
	Description	435	149	99	67	50%
	Hybrid Keywords	540	78	40	92	67%
	Relative	240	189	168	153	47%
	Coordinates	562	100	60	28	51%
	Hybrid Spatial	592	56	51	51	62%
	Hybrid Approach	668	23	10	50	79%

•Name:
TP: The system displays the result of 93 sensors from 120 sensors with the name "contact temperature sensor."
TN: The system does not display the result of 214 sensors, which is not a "contact temperature sensor."
FP: The system displays 416 sensors as a contact sensor not named "contact temperature sensor" out of the remaining 630 sensors.
FN: The system does not display the 27 sensors from 120 sensors named "contact temperature sensor."
•URI:
TP: Out of 750 sensors, the system displays the result of 420 sensors from URI.
TN: The system does not display the results of 99 sensors which do not match with the URI from the remaining 330 sensors.
FP: The system displays the result of 62 sensors that do not match the URI from the remaining 330 sensors.
FN: The system does not display the result of 169 sensors that matched the URI out of the remaining 330 sensors.
•Type:
TP: The system displays the result of 307 sensors from 480 with type temperature.
TN: The system does not display the results of 86 sensors whose type is not temperature out of the remaining 270 sensors.
FP: The system displays the results of 184 sensors whose type is not temperature out of 270 remaining sensors.
FN: The system does not display the results of 173 sensors from 480 sensors whose type is temperature.
•Description:
TP: Out of 750 sensors, the system accurately displays the result of 435 sensors of the given description.
TN: The system does not display the results of 67 sensors whose description does not match the query out of the remaining 315 sensors.
FP: The system displays the results of 149 sensors whose description does not match with the query out of the remaining 315 sensors.
FN: The system does not display the result of 99 sensors described in a description out of the remaining 315 sensors.

•Hybrid Keywords:
TP: Out of 750 sensors, the system accurately displays the result of 540 sensors using the hybrid keywords approach.
TN: The system does not display the results of 92 sensors out of the remaining 210 sensors whose location has been changed.
FP: The system displays the results of 78 sensors that do not match the query out of the remaining 210 sensors.
FN: The system does not display the results of 40 sensors matched with the query out of the remaining 210 sensors.
•Relative Location:
TP: Out of 750 sensors, the system displays the result of 240 sensors by entering the city, country, state, and street.
TN: The system does not display the result of 153 sensors out of the remaining 510 whose location has been changed.
FP: The system displays the result of 189 sensors that do not match the query from the remaining 510 sensors.
FN: The system does not display the results of 168 sensors that matched the query out of the remaining 510 sensors.
•Coordinates:
TP: Out of 750 sensors, the system displays the result of 562 sensors accurately by using Latitude and Longitude.
TN: The system does not display the results of 28 sensors out of 188 sensors that do not match the query.
FP: The system displays the results of 100 sensors that do not match the query from the remaining 188 sensors.
FN: The system does not return the results of 60 sensors that matched the query out of the remaining 188 sensors.
•Hybrid Spatial:
TP: Out of 750 sensors, the system accurately displays the result of 592 sensors using a hybrid spatial approach.
TN: The system does not display the results of 51 sensors out of the remaining 158 sensors, which does not match the query.
FP: The system displays the results of 56 sensors that do not match the query from the remaining 158 sensors.
FN: The system does not display the results of 51 sensors matched with the query out of the remaining 158 sensors.
•Hybrid Approach:
TP: Out of 750 sensors, the system displays the result of 668 sensors by using a hybrid approach.
TN: The system does not display the results of 50 sensors out of the remaining 82 sensors, which do not match the query.
FP: The system displays the results of 23 sensors that do not match the query from the remaining 82 sensors.
FN: The system does not display the results of 10 sensors matched with the query out of the remaining 82 sensors.

6. Discussion

After analyzing the results, if the user wants to index the “things” based on a single keyword, the user must consider whether to focus on the response time or accuracy. In the case of

keywords, if the user has to focus on the response time, we propose using a name keyword for indexing that has the best response time.

Based on a single keyword, if a user has to focus on accuracy, we propose using a URI because the accuracy of a URI is 57%, which is better than other keywords. If a user has to focus on response time, we discourage using a description keyword with the worst response time. If a user has to concentrate on accuracy, we discourage using a Type keyword with 48% accuracy, which is less than other keywords.

In the case of spatial, if the user has to focus on the response time, we propose using a relative location to index the best response time. If a user's center of interest is accuracy, we suggest using coordinates because the accuracy of coordinates is 51%, which is better than relative location but with the worst response time.

If a user has to focus on the accuracy, we discourage using a relative location with 47% accuracy, more diminutive than coordinates. If a user has to focus on the response time, we propose using a hybrid approach (Hybrid keywords and Hybrid Spatial) for indexing, which has the best response time than hybrid keywords and hybrid spatial. Suppose a user has to focus on accuracy. We propose using a hybrid approach because the accuracy of a hybrid approach is 79%, which is better than hybrid keywords and hybrid spatial individually.

In general, we propose to use a hybrid indexing approach that has the best results concerning response time and accuracy.

7. Conclusion

Search engines use indexing and ranking mechanisms for searching. The existing search engines such as Google, Yahoo, and ask.com can efficiently work to search textual documents. These search engines perform both indexing as well as the ranking of data. In the case of searching the physical objects, such search engines do not work because these search engines only work for keywords-based queries. However, few search engines exist for exploring physical things, such as Dyser, Snoogle, and ThingSeek. All of these have limitations for finding dynamically changing devices. In this paper, the authors have developed a searching algorithm and implemented a system for dynamic searching and efficient indexing of Things. In this paper, the authors have used Keyword based, spatial, and hybrid approaches for indexing a Thing. In this paper, the authors have made the schema according to W3C JSON-LD and indexed the description of the Thing according to that model in Solr. Authors have evaluated their approach by analyzing the search query response timings and the accuracy of their search results. Authors have used different indexing approaches, such as keywords-based, spatial, and hybrid approaches. Results indicate that response time indexing accuracy is better with the hybrid approach than with keyword-based and spatial indexing. In the future, we will make it more efficient by providing a better ranking mechanism**.**

References

[1] Y. Qin, Q. Z. Sheng, N. J. Falkner, S. Dustdar, H. Wang, and A. V. Vasilakos, "When things matter: A data-centric view of the internet of things," *arXiv preprint arXiv:1407.2704*, 2014.

[2] D. Guinard, "What is the Web of Things?," Web of Things, Apr. 8, 2017. [Online]. Available: https://webofthings.org/2017/04/08/what-is-the-web-of-things/

[3] M. R. Faheem, T. Anees, and M. Hussain, "The Web of Things: Findability Taxonomy and Challenges," *IEEE Access*, vol. 7, pp. 185028-185041, 2019. Article (CrossRef Link)

[4] P. Morville, *Ambient findability: What we find changes who we become*, O'Reilly Media, Inc., 2005. Article (CrossRef Link)
[5] B. J. Jansen, D. L. Booth, and A. Spink, "Determining the user intent of web search engine queries," in *Proc. of the 16th international conference on World Wide Web* (*WWW'07)*, Banff, Canada, pp. 1149–1150, 8-12 May 2007. Article (CrossRef Link)
[6] Apache Lucene. [Online]. Available: https://lucene.apache.org/
[7] V. Charpenay, "Json-schema". [Online]. Available: https://www.w3.org/2019/wot/json-schema
[8] S. Kaebisch, "Things description". [Online]. Available: https://www.w3.org/TR/wot-thing-description
[9] D. Lewandowski, "The retrieval effectiveness of web search engines: considering results descriptions," *Journal of Documentation*, vol. 64, no. 6, pp. 915-397, 2008. Article (CrossRef Link).
[10] D. Lewandowski, "Web searching, search engines and information retrieval," *Information Services & Use*, vol. 25, no. 3-4, pp. 137–147, 2005. Article (CrossRef Link).
[11] L. Paolino, M. Sebillo, G. Tortora, and G. Vitiello, "Towards a new approach to query search engines: the search tree visual language," *Software: Practice and Experience*, vol. 40, no. 8, pp. 735–750, 2010. Article (CrossRef Link).
[12] M. Liu, T. Leppanen, E. Harjula, Z. Ou, A. Ramalingam, M. Ylianttila, and T. Ojala, "Distributed resource directory architecture in machine-to-machine communications," in *Proc. of 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Lyon, France, pp. 319–324, 7-9 Oct 2013. Article (CrossRef Link)
[13] A. J. Jara, P. Lopez, D. Fernandez, J. F. Castillo, M. A. Zamora, and A. F. Skarmeta, "Mobile digcovery: discovering and interacting with the world through the internet of things," *Personal and Ubiquitous Computing*, vol. 18, no. 2, pp. 323–338, 2014. Article (CrossRef Link)
[14] N. Hussain, T. Anees, and AzeemUllah, "Development of a Novel Approach to Search Resources in IoT," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 9, no. 9, pp. 385-398, 2018. Article (CrossRef Link)
[15] L. Sciullo, C. Aguzzi, M. Di Felice, and T. S. Cinotti, "Wot store: Enabling things and applications discovery for the w3c web of things," in *Proc. of 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, pp 1-8, 11-14 Jan 2019. Article (CrossRef Link)
[16] A. G. Khan, A. H. Zahid, M. Hussain, M. Farooq, U. Riaz, and T.M. Alam, "A journey of WEB and Blockchain towards the Industry 4.0: An Overview," in *Proc. of 2019 International Conference on Innovative Computing (ICIC)*, Lahore, Pakistan, pp. 1-7, 1-2 Nov 2019. Article (CrossRef Link)
[17] M. W. Nadeem, H. G. Goh, M. Hussain, M. Hussain, and M. A. Khan, "Internet of Things for Green Building Management: A Survey," in *Role of IoT in Green Energy Systems*, IGI Global, 2021, pp. 156-170. Article (CrossRef Link)
[18] T.-A. Hoang-Vu, H. T. Vo, and J. Freire, "A unified index for spatio-temporal keyword queries," in *Proc. of the 25th ACM International Conference on Information and Knowledge Management (CIKM'16)*, Indianapolis, USA, pp. 135-144, 24-28 Oct 2016. Article (CrossRef Link)
[19] A. Khodaei, C. Shahabi, and A. Khodaei, "Temporal-textual retrieval: Time and keyword search in web documents," *International Journal of Next-Generation Computing*, vol. 3, no. 3, pp. 288-312, 2012. Article (CrossRef Link)
[20] S. Nepomnyachiy, B. Gelley, W. Jiang, and T. Minkus, "What, where, and when: keyword search with spatio-temporal ranges," in *Proc. of the 8th Workshop on Geographic Information Retrieval (SIGSPATIAL '14)*, Dallas, Texas, pp. 1– 8, 4-7 Nov 2014. Article (CrossRef Link)
[21] B. Ostermaier, K. Romer, F. Mattern, M. Fahrmair, and W. Kellerer, "A real-time search engine for the web of things," in *Proc. of 2010 Internet of Things (IOT)*, Tokyo, Japan, pp. 1–8, 29 Nov-1 Dec 2010. Article (CrossRef Link).

[22] H. Wang, C. C. Tan, and Q. Li, "Snoogle: A search engine for pervasive environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 8, pp. 1188–1202, 2010. Article (CrossRef Link)
[23] I. Nadim, Y. Elghayam, and A. Sadiq, "Semantic discovery architecture for dynamic environments of web of things," in *Proc. of 2018 International Conference on Advanced Communication Technologies and Networking (CommNet)*, Marrakech, Morocco, pp. 1–6, 2-4 Apr 2018. Article (CrossRef Link)
[24] A. Shemshadi, Q. Z. Sheng, Y. Qin, A. Sun, W. E. Zhang, and L. Yao, "Searching for the internet of things: where it is and what it looks like," *Personal and Ubiquitous Computing*, vol. 21, no. 6, pp. 1097–1112, 2017. Article (CrossRef Link).
[25] M. Zhou and Y. Ma, "A web service discovery computational method for iot system," in *Proc. of 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems(CCIS)*, Hangzhou, China, pp. 1009– 1012, 30 Oct-1 Nov 2012. Article (CrossRef Link)
[26] Z. Li, K. C. Lee, B. Zheng, W. C. Lee, D. Lee, and X. Wang, "Ir-tree: An efficient index for geographic document search," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 4, pp. 585–599, 2011. Article (CrossRef Link)
[27] Y. Zhou, S. De, W. Wang, and K. Moessner, "Search techniques for the web of things: A taxonomy and survey," *Sensors*, vol. 16, no. 5, p. 600, 2016. Article (CrossRef Link).
[28] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub et al., "The geojson format," *Internet Engineering Task Force (IETF)*, 2016. Article (CrossRef Link)
[29] A. Ciortea, S. Mayer, S. Bienz, F. Gandon, and O. Corby, "Autonomous search in a social and ubiquitous Web," *Personal and Ubiquitous Computing*, pp.1-14, 2020. Article (CrossRef Link)
[30] L. Sciullo, L. Gigli, A. Trotta, and M. Di Felice, "WoT Store: Managing resources and applications on the web of things," *Internet of Things*, vol. 9, pp. 100164, 2020. Article (CrossRef Link)
[31] A. U. Rehman, M. Hussain, M. Idress, A. Munawar, M. Attique, F. Anwar, and M. Ahmad, "E-cultivation using the IoT with Adafruit cloud," *International Journal of Advanced and Applied Sciences*, vol. 7, no. 9, pp. 75-82, 2020. Article (CrossRef Link)
[32] W. I. Grosky, A. Kansal, S. Nath, J. Liu, and F. Zhao, "Senseweb: An infrastructure for shared sensing," *IEEE multimedia*, vol. 14, no. 4, pp. 8–13, 2007. Article (CrossRef Link)
[33] K. K. Yap, V. Srinivasan, and M. Motani, "Max: human-centric search of the physical world," in *Proc. of the 3rd international conference on Embedded networked sensor systems* (*SenSys05*), San Diego, California, USA, pp. 166–179, 2-4 Nov, 2005. Article (CrossRef Link)
[34] P. Aswale, A. Shukla, P. Bharati, S. Bharambe, and S. Palve, "An overview of internet of things: architecture, protocols and challenges," in *Information and Communication Technology for Intelligent Systems*, vol. 106, Smart Innovation, Systems and Technologies, Satapathy, S., Joshi, A. (eds), Singapore: Springer, 2019, pp. 299–308. Article (CrossRef Link)
[35] N. K. Tran, Q. Z. Sheng, M. A. Babar, and L. Yao, "Searching the web of things: State of the art, challenges, and solutions," *ACM Computing Surveys (CSUR)*, vol. 50, no. 4, pp. 1–34, 2018. Article (CrossRef Link).
[36] L. Sciullo, C. Aguzzi, M. Di Felice, and T. S. Cinotti, "Wot store: Enabling things and applications discovery for the w3c web of things," in *Proc. of 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, pp. 1-8, 11-14 Jan 2019. Article (CrossRef Link)
[37] A. Shemshadi, Q. Z. Sheng, and Y. Qin, "Thingseek: A crawler and search engine for the internet of things," in *Proc. of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval* (*SIGIR '16*), Pisa, Italy, pp. 1149–1152, 17–21 July, 2016. Article (CrossRef Link)
[38] Z. Ding, Z. Chen, and Q. Yang, "Iot-svksearch: a real-time multimodal search engine mechanism for the internet of things," *International Journal of Communication Systems*, vol. 27, no. 6, pp. 871–897, 2014. Article (CrossRef Link)

[39] GitHub, Xml document parser for php and laravel. [Online]. Available: https://github.com/orchestral/parser
[40] GitHub, Json-ld processor for php. [Online]. Available: https://github.com/lanthaler/JSONLD

Muhammad Rehan Faheem was born in Pakistan in 1990. He received his BSIT degree from The Islamia University of Bahawalpur, Pakistan, in 2012 and the MPhil degree in Computer Science from NCBAE, Lahore, Pakistan, in 2016. Currently he is doing PhD from University of Management & Technology Lahore, Pakistan, under the supervision of Dr. Muzammil Hussain and Co-supervision of Dr. Tayyaba Anees. His research interests are Internet of Things, Web of Things, Image Processing, Information retrieval, Computer graphics, computer vision, graphics modelling and Machine Learning.

DR. TAYYABA ANEES received her Ph.D. degree from the Vienna University of Technology, Vienna, Austria, in 2012. Her Ph.D. dissertation is in the area of service-oriented architecture and web services availability domain. She has worked as the Project Assistant at Vienna University of Technology for four years. She is currently working as the Program Head Software Engineering/Assistant Professor at the Software Engineering Department, School of Systems and Technology, University of Management and Technology, Lahore. Her research interests include Service-oriented architecture, Web services, Web of Things (WOT), Semantic Web, Software availability, Software Safety, Software Fault tolerance and real-time data warehousing.

DR. Muzammil Hussain is working as an Assistant Professor in the Department of Computer Science, School of Systems and Technology, University of Management and Technology, Lahore, Pakistan. He completed his Ph.D. from the University of Malaya, Kuala Lumpur, Malaysia (and he got Bright Sparks Program scholarship from University of Malaya for his PhD studies). He received his Bachelor of Science in Computer Science from COMSATS University Islamabad in 2013, and awarded with Gold Medal. He has served as a Research Associate for more than three years (2013-2017) in the Department of Computer Systems and Technology, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia. He led or member for many funded research projects and he has published more than 15 research articles in prestigious international conferences and journals. His potential research areas include, Web services, Web of Things, network security, FinTech Security, Blockchain, Android Security, Bioinformatics, IoT Security, SDN Security, and Internet Security.