**Journal of Engineering and Sustainable Development**
Vol. 29, No. 03, May 2025
https://jeasd.uomustansiriyah.edu.iq/index.php/jeasd
ISSN 2520-0917
**Research Article**
https://doi.org/10.31272/jeasd.2723

# A Proposed Mechanism for Trusted Evaluation of Web Services

Aya Joumaa*, Basel Hasan, Nasser Nasser

Department of Software and Information Systems, Faculty of Informatics Engineering, Tishreen University, Latakia, Syria

*Email: aya.joumaa@tishreen.edu.sy

**Article Info**

| | |
|---|---|
| **Received** | **25/05/2024** |
| **Revised** | **06/01/2025** |
| **Accepted** | **20/01/2025** |

**Abstract**

Web services are one of the important innovations in the software field, and they are used on a large scale in modern software systems. Web services have evolved and significantly increased during the last two decades, resulting in large numbers of similar services in terms of function and can be used to perform the same tasks. As a result, choosing the most appropriate web service to meet users' needs has become an important research topic. This research presents a trusted mechanism for evaluating web services by relying on Quality of Service (QoS) metrics, where several metrics that are measurable on the user's side were chosen. Work has been done to make the mechanism general, scalable, and expandable to support additional quality metrics, in addition to focusing on the ease of using this mechanism and the evaluation calculation for the different web services. This research works on ensuring the reliability of the results when sending them by signing results digitally. To prove the possibility of implementing the proposed mechanism, it was tested on three groups of web services, with each group consisting of three functionally similar services. The experiment demonstrated the possibility of implementing the proposed mechanism on different web services.

**Keywords:** Digital signature; Quality of Service metrics; Service-Oriented Architecture; Web services evaluation.

## 1. Introduction

Web services are software systems that perform a task or a set of tasks, deployed and called over the World Wide Web. Web services are designed basically to allow variant systems to communicate with each other and exchange data across the internet, regardless of the frameworks or programming languages used to build these systems. Web services define the rules for communication between different systems, and to achieve this, they use standard protocols and architectural styles to ensure compatibility and interoperability. The most popular standards used in web services are Simple Object Access Protocol (SOAP) and Representational State Transfer architectural style (REST) [1],[2].

The web has developed, and its popularity has increased over the last few decades. Many companies have taken it to the web to present their services, which led to the development of web services and increased their numbers significantly. On the other hand, given that many services may be similar to each other and provide the same functions, efforts were made to find mechanisms that help users choose the appropriate services for them, so the evaluation of web services has taken a large portion of the researchers' interest [3],[4].

The evaluation of web services is the process of analyzing and studying the various attributes of services to determine the quality of the service or the extent to which it meets consumers' requirements and the functions desired from it. The attributes of services can be classified into functional or behavioral attributes and non-functional or non-behavioral attributes. On one hand, functional attributes define what the service must do and the behaviors it must exhibit. On the other hand, non-functional attributes determine the quality of service (QoS), and they are represented by a set of quality-of-service metrics that are studied, such as response time, maximum throughput, and availability [1],[5].

## 2. Related Work

The research that studied the mechanisms for evaluating and recommending web services has branched out, and many strategies have been followed. Some of the research has studied the evaluation of web services based on quality of service (QoS) metrics, given that the quality of service describes its capabilities to meet the requirements of consumers in a specific field. Zhang et al. [6] proposed a tool to measure web service quality independent of service providers or consumers. The proposed tool provides a dynamic approach to measure and store QoS metrics, where the similarity is calculated between the QoS values announced by service providers and the actual values that were calculated; similarity is used to evaluate the reputation level of the web service. Bouasker et al. [7] introduced a monitoring system whose role is to assess and track the values of web service quality standards to ensure new information about the quality of service. The system is based on QoS metrics, where these metrics are measured, and the evaluation of services is calculated according to users' preferences.

Another section of research is concerned with studying the quality of service standards and data quality standards, given the importance of data quality and its recent impact on some services, such as medical services. Romdhani et al. [8] presented a model for evaluating service quality and determining the degree of trust in it so that this model integrates a set of service quality indicators, including service performance and data quality. The presented model is based on the fact that it is necessary to determine to what extent the service can guarantee a certain level of quality. However, it is also necessary to determine the extent to which the data provided can be trusted for data services. Therefore, the trust measure for the service can be an indicator that represents both the quality of the service and data quality. Song et al. [9] studied a framework for measuring the quality of service called Quality of Information (QoI), which is a quality measure complementary to quality of service that measures the degree to which web services meet non-functional requirements related to data, where four measures were studied: Accuracy, which measures the quality of the results whether they are correct or meet the user's expectations, Completeness which define the level of missing information in the results. Coverage Indicates how the web service interacts with various inputs. Freshness measures the number of times the web service interface related to the data updates its data from the data source.

Some research has focused on examining user experience characteristics and the popularity of the services for consumers. Li et al. [10] proposed an evaluation approach based on users' experience characteristics, such as click-through rate and the number of components generated by web service constructs. Where work was done to build an evaluation system based on users' experience instead of relying on some characteristics of Quality of Service (QoS) standards such as response time, reliability, and other metrics. Wu et al. [11] suggested a method to recommend web services called the popularity-aware and diverse method of web API compositions' recommendation (PD-WACR), where the web services' popularity and compatibility are modeled with an API correlation graph. After that, correlation graph-based web APIs' recommendation is implemented with guaranteed popularity and compatibility.

In this research, we designed and developed a trusted mechanism for evaluating web services, aiming to make the mechanism scalable and expandable, enabling it to work on various common web service types, namely SOAP and RESTful web services. During the design of the mechanism, an evaluation strategy was followed based on QoS standards, where five metrics were considered: service response time, productivity, availability, accessibility, and successability. The calculation of evaluations for all services has also been scheduled periodically through the use of scheduled cron jobs, in addition to ensuring the authority of the evaluation results sent for services by adding a digital signature to the results when they are sent. The mechanism is provided through a website so that various services can be added and ratings calculated dynamically, in addition to displaying the added services and their ratings to end users.

## 3. Materials and Methods

This section describes the methodologies used in designing the proposed evaluation mechanism. It consists of eight subsections: (1) Service-Oriented Architecture (SOA), (2) The architecture of internal and external web services, (3) SOAP web services, (4) RESTful web services, (5) Digital Signature, (6) Techniques used in developing the proposed mechanism, (7) Design the proposed mechanism, (8) Presented the proposed mechanism.

### 3.1. Service-Oriented Architecture (SOA)

The SOA design model is a set of components whose interface descriptions can be published, discovered, and called across a network. These components are provided as independent services that can be accessed uniformly. SOA provides an infrastructure that facilitates the discovery and use of services while maintaining loose coupling between service providers and consumers. Fig. 1 shows the provided infrastructure by SOA, which comprises a service provider that hosts services and publishes their interfaces. A service broker is used to publish descriptions of service interfaces and all necessary means for accessing services. Service requesters are the consumers who use the service [12].

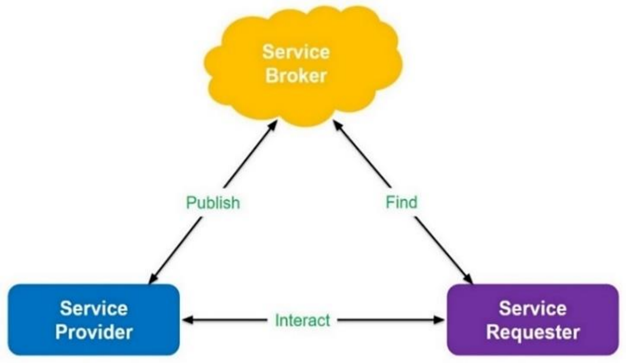


**Figure 1.** Infrastructure provided by SOA.

**Updated from:** Sunyaev, A., (2020). Web services, in: Internet Computing [12].

The concept of a web service is closely related to a Service-Oriented Architecture, where SOA can be implemented through web services technologies. Web services do not need to be deployed in the SOA environment, and SOA does not need to rely on web services technologies. SOA can be achieved using almost any programming model, but it will be difficult to achieve the required degree of loose coupling without using web services [12]. The two most common web service types are SOAP and RESTful web service [13].

### 3.2. The Architecture of Internal and External Web Services

The internal architecture of a web service describes a complex and multi-level system consisting of several layers that communicate with each other, as represented in Fig. 2. Initially, the web service interface level receives incoming messages. It passes them to a program that translates them into a format that the middleware layer understands. It then sends the messages to the middleware layer. Middleware, in turn, interacts with the web service's internal processes and resources. The service interface is implemented and described using standard technologies such as SOAP and REST. These two standards are closely related to web services, where services that implement their interfaces using SOAP are called SOAP web services, while those that implement their interfaces using REST are called RESTful web services [12].

The external web service architecture facilitates access to internal services whose respective functions are exposed through their web service interface. The external architecture can be described as a program that acts as an intermediary between service providers and service consumers and is in the form of a central service broker. The role of a service broker depends on how you implement it [12].

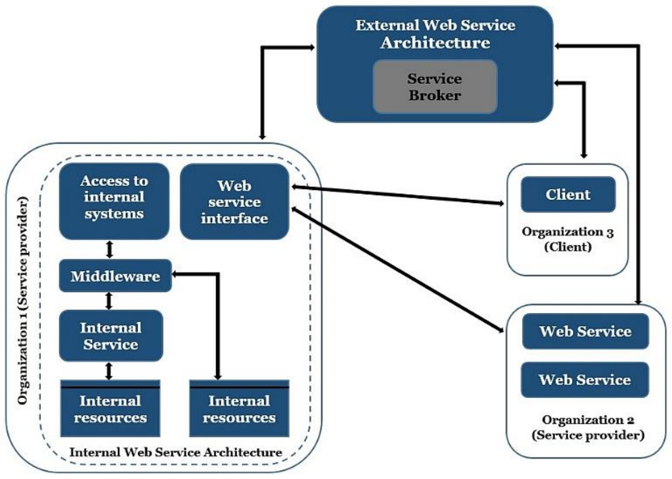
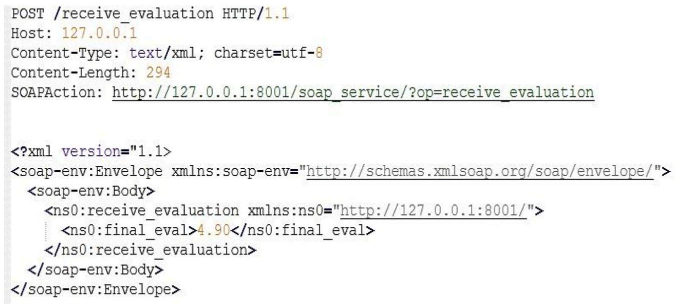

External Web Service Architecture
Service Broker
Access to internal systems
Web service interface
Middleware
Internal Service
Internal resources
Internal resources
Organization 1 (Service provider)
Internal Web Service Architecture
Client
Organization 3 (Client)
Web Service
Web Service
Organization 2 (Service provider)

**Figure 2.** The internal and external architecture of web services.

**Updated from:** Sunyaev, A., (2020). Web services, in: Internet Computing [12].

### 3.3. SOAP Web Services

Web services whose interfaces are implemented using SOAP are known as SOAP web services. SOAP is a protocol used to exchange data, where data is sent in the form of messages written in XML, known as SOAP messages; Fig. 3 shows an example of a SOAP message for a client request. Several network protocols can be used to transfer SOAP messages. Still, the Hypertext Transfer Protocol (HTTP) is the most popular protocol for web services and is commonly used for sending SOAP messages. A SOAP web service publishes its functionality through machine-readable service descriptions based on the Web Services Description Language (WSDL) [14],[15].



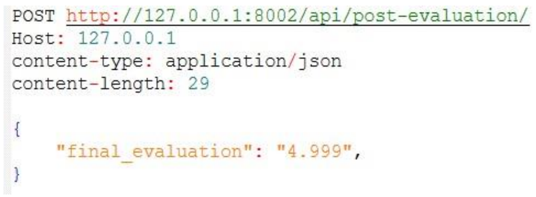
```
POST /receive_evaluation HTTP/1.1
Host: 127.0.0.1
Content-Type: text/xml; charset=utf-8
Content-Length: 294
SOAPAction: http://127.0.0.1:8001/soap_service/?op=receive_evaluation


<?xml version="1.1>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
  <soap-env:Body>
    <ns0:receive_evaluation xmlns:ns0="http://127.0.0.1:8001/">
      <ns0:final_eval>4.90</ns0:final_eval>
    </ns0:receive_evaluation>
  </soap-env:Body>
</soap-env:Envelope>
```

**Figure 3.** SOAP message for a client request.

### 3.4. RESTful Web Services

Web services whose interfaces are implemented using the REST architectural style are known as RESTful web services. REST describes an architectural style created to represent and organize distributed systems. It helps in designing loosely coupling applications. REST can be defined as a set of constraints that must be taken into account when developing applications. Constraints focus on how different components communicate in the system rather than on the semantics of these components. Therefore, RESTful web services are designed with the constraints of the REST architectural style applied and based on the HTTP transport protocol. Fig. 4 shows an example of a REST message for a client request [16],[17].



```
POST http://127.0.0.1:8002/api/post-evaluation/
Host: 127.0.0.1
content-type: application/json
content-length: 29

{
    "final_evaluation": "4.999",
}
```

**Figure 4.** REST message for a client request.

### 3.5. Digital Signature

The digital signature is considered one of the most common forms of electronic signature. It is defined as information in electronic form related to the data message, and it may be included or added to the message or logically linked to it. The digital signature is used to assert the identity of the sender of the message and to indicate his agreement with the information contained in it. A digital signature helps verify the message's authentication and integrity and ensures nonrepudiation. Digital signature is achieved through asymmetric cryptographic algorithms, where the message is signed using the sender's private key, and the signature is verified on the recipient's side using the sender's public key. There are several mechanisms to achieve digital signatures, as with short messages, the message is signed directly.

Meanwhile, with long messages, only the hash of the message is signed; this is because asymmetric encryption algorithms take a long time to sign long messages. To avoid this, the hash of the message is formed, and then it is signed. Many algorithms are used with digital signatures, one of the most prominent of which is the RSA algorithm, the ElGamal algorithm, and the DSA algorithm [18].

RSA is one of the popular asymmetric algorithms that contains three phases; the first phase is the generation of public and private keys used during the encryption and decryption processes. The next phase is the encryption process, which transforms the plaintext into ciphertext. The third phase is decryption, which includes decrypting the ciphertext and converting it into plaintext on the receiver side [19],[20]. RSA is based on the difficulty of factorizing a large number; the public key is produced from two numbers, one of which is the outcome of multiplying two large prime numbers, and these two primes are used to construct the secret code. Therefore, the secret key will be compromised if an adversary successfully factors a large number. For this reason, the encryption strength relies on the key's size, and the doubling or tripling of the key size leads to an exponential increase in the strength of encryption. The standard RSA key's length is 2048 bits; experts expect this length will be compromised shortly. However, this task seems unachievable at this time [21].

### 3.6 Techniques Used in Developing the Proposed Mechanism

*3.6.1 Wagtail CMS* [1]

Wagtail is an open-source content management system written in Python and built on the Django web framework. Its modular architecture allows it to expand and customize its functionality to suit the needs of users and developers, whether by adding new features or integrating with third-party services. Wagtail provides an easy-to-use interface with many features that help users add and track website content.

*3.6.2 Zeep Library* [2]

Zeep is a Python library that provides fast, modern functionality for SOAP clients; Zeep parses WSDL documents and generates the code needed to use the services and operations within the document. In addition, Zeep facilitates sending requests to SOAP web services and analyzing the server response to get results.

*3.6.3 Cron Jobs* [3]

Cron jobs are Linux operating system commands used to schedule tasks to be executed at some point in the future. They are commonly used to schedule tasks to be executed periodically according to a specific timetable written in Cron format.

*3.6.4 SignXML Library* [4]

SignXML is a library that implements the W3C standard for the digital signing of XML files, known as W3C XML Signature. The library is implemented in the Python programming language. It helps create a digital fingerprint for XML files, digitally sign them, and verify the signature.

*3.6.5 PyCryptodome Library* [5]

PyCryptodome is a stand-alone Python package that covers low-level encryption basics. It performs many tasks, including creating a fingerprint for files, creating and verifying the digital signature of messages, encrypting data, and decrypting data.

*3.6.6 OpenSSL Library* [6]

OpenSSL is a software library for various applications that provides secure communications over computer networks. It is widely used to allow users to perform SSL-related tasks, such as generating private keys and creating Certificate Signing Requests (CSRs).

### 3.7 Design the Proposed Mechanism

This research took advantage of the evaluation mechanism proposed by Taycir Bouasker, Mahjoub Langar, and Riadh Robbana [7]. The reference introduces several algorithms to design a mechanism for evaluating web services, considering the client's requirements, where communication occurs continuously between clients, and the proposed mechanism in the reference [7] for calculating evaluations. In this research, proposed algorithms are analyzed and allocated to design an evaluation mechanism directed toward web service users and providers.

The proposed mechanism calculates evaluations for web services periodically to ensure the freshness and reliability of the evaluations. The mechanism collects all similar services together and arranges them according to the evaluation results while giving users the ability to filter services according to the priority of the factors studied for them, thus helping them choose the appropriate service for them according to their different preferences, providing users with graphic charts that help them know the development of services and evaluating them over the years, in addition to ensuring the reliability of the evaluations sent for web services by signing them digitally.

*3.7.1 Define the evaluation metrics*

The proposed mechanism follows the evaluation strategy based on Quality of Service (QoS) metrics, where five metrics are selected to be measurable from the user side; they are [7]:

- **Response Time (RT):** The period required from the send request time until the response is received.

- **Maximum Throughput (TH):** This metric refers to the number of requests processed during the time unit. It is determined mathematically according to the equation:

$$TH = \frac{ProcessedRequestsInTimeUnit}{\text{TimeUnit}} \quad (1)$$

- **Availability (AV):** It determines the availability of the web service, as it specifies the time during which the web service is available during the measured time. It is defined mathematically according to the equation:

[1]*Wagtail CMS - Django Content Management System*
[2]*Zeep: Python SOAP client — Zeep 4.1.0 documentation*
[3]*Free cronjobs - from minutely to once a year. - cron-job.org*
[4]*SignXML documentation*
[5]*PyCryptodome's documentation*
[6]*OpenSSL - openssl.org*

$$AV = \frac{\text{TimeServiceAvailable}}{\text{MeasuringTime}} \quad (2)$$

- **Accessibility (ACC):** This metric defines the accessibility of a web service. It is determined mathematically according to the following equation:

$$ACC = \frac{NumberOfAckMessages}{\text{NumberOfRequests}} \quad (3)$$

- **Successability (SUCC):** It measures the ability to process requests successfully and return responses to the client. It is determined mathematically according to the following equation:

$$SUCC = \frac{NumberOfResponses}{\text{NumberOfRequests}} \quad (4)$$

In this research, all service operations are considered when calculating the metrics. Therefore, all service operations and the information necessary to send requests to them correctly are extracted. To calculate any metric, its value is measured for each operation, and the average represents the value of this metric for the service.

Each metric was determined through a set of five parameters whose value is determined by the client, which are [7]:

- The class (C) determines the importance of the metric when calculating the service evaluation. It may take one of the following three values: N means the metric is mandatory, DNN means it is preferable to calculate the metric, but it is not mandatory, and O means it is not mandatory (optional).
- The weight (W): It is the relative importance of a metric in comparison to other metrics.
- QoS attribute (Q): The metric that is calculated.
- Best Value (BV): Defines the best value a customer can expect for the metric.
- Worst Value (WV): Specifies the worst value of the metric for the customer.

The parameters within the research are identified according to the following:

- Class (C): Its value is set to N for all metrics, so the five metrics will be used to calculate the evaluation of all web services.
- Worst Value (WV) and Best Value (BV): These values are variable and manageable through the control panel for each category.
- Weight (W): It is controlled in a variable and manageable through the control panel for each category. The sum of all metrics' weights must be ten because the evaluations are calculated out of ten.

Therefore, evaluations are calculated for all services in the same category according to the same parameters' values to compare them effectively.

Initially, WV, BV, and W parameters are given the default values defined in Table 1.

**Table 1.** Default values for the BV, WV, and W parameters.

| Metric | BV | WV | W |
|---|---|---|---|
| RT | 1 Sec | 5 Sec | 1.4 |
| TH | 6 Req/Sec | 1 Req/Sec | 1.4 |
| AV | 100 | 50 | 2.8 |
| ACC | 100 | 70 | 2.8 |
| SUCC | 100 | 70 | 1.6 |

*3.7.2 Calculating the evaluation value*

After calculating the metrics and determining their five parameters, the web services evaluation is calculated by following several algorithms [7]. In this research, the algorithms were customized, and only two algorithms were used, which are:

- **Partial score computation algorithm:** Partial scores for metrics are calculated by calculating the distance between the previously measured metric value (MV) and the range of acceptable metrics values, BV and WV [7].

**Algorithm 1:** Partial score computation algorithm [7]

```
Input: A requirement defined by (C, W, Q, BV, WV, MV)
1: if ((($Q \in Q+$) AND ($MV \geq BV$)) OR (($Q \in Q-$) AND ($MV \leq BV$))) then ▷ Best Case
2:     $score_r \leftarrow 1$
3: else if ((($Q \in Q+$) AND ($MV \leq WV$)) OR (($Q \in Q-$) AND ($MV \geq WV$))) then ▷ Worst Case
4:     $score_r \leftarrow 0$
5: else if ($Q \in Q+$) then ▷ Case of positive attributes with accepted values
6:     $score_r \leftarrow 100 * ((MV - WV) / (BV - WV))$ ▷ How far is the MV compared to WV
7: else ▷ Case of negative attributes with accepted values
8:     $score_r \leftarrow 100 * ((WV - MV) / (WV - BV))$
9: end if
```

- **Global score computation algorithm:** The final evaluation is initialized with zero. Then, for each metric, its partial points calculated in the previous algorithm are multiplied by the parameter value (W) and added to the final evaluation value [7].

**Algorithm 2:** Global score computation algorithm [7]

**Input:** The table of all requirements defined by *{(C[i], W[i], Q[i], BV[i], WV[i], MV[i]); i=1...M}* the table of all requirements scores defined by *{score$_r$[i]; i = 1...M}*

```
1: WSScore ← 0    ▷ WSScore: the global score assigned to
   the WS initialized to 0

2: for i := 1 → M do    ▷ M is the number of all
   Requirements

3:     WSScore ← WSScore + W [i] ∗ scorer [i]    ▷ scorer
       [i] is the partial score of the ith requirement

4: end for
```

Within the research, the evaluation of web services is calculated periodically to ensure the evaluations are up-to-date and objective. The previous steps are performed once every 6 hours a day, achieved using cron jobs.

The concept of service quality is still under study, and no unified definition exists. Instead, there are many different points of view on how to define it. Some may take product quality as a basis for determining service quality. In contrast, others may define quality based on user preferences, whereas some resort to using the concept of service value to infer quality. In contrast, others calculate service quality based on the purpose and context for which the service is used [22]. Therefore, when designing a system to evaluate web services based on service quality metrics, we cannot create a general and unified model because there is no fixed concept of service quality but rather a set of interconnected approaches [23]. In the research paper [7], the previous metrics and algorithms were used to calculate the evaluation of web services by relying on user preferences to calculate service quality. Therefore, to evaluate a service, the user will communicate with the proposed evaluation mechanism and determine the values of the four parameters (C, BV, WV, and W) for each quality metric. In addition, for each metric, the user has to determine the technical attributes necessary to measure its value, such as test start time, test end time, frequency, and iteration. The values of the metrics are estimated based on the test periods and technical attributes specified by the user. The measured values are used with the two previous algorithms to calculate the final evaluation by relying on the values of the parameters (C, BV, WV, W) specified by the user for each metric. As for this research, the previous metrics and algorithms were used to develop a mechanism for evaluating the quality of web services based on the context of service use and its purpose. According to the proposed mechanism, many web services can be added and classified into different groups based on the context of service use and the functions they provide. Therefore, for all services that belong to the same category (used in the same context), the values of the parameters (C, BV, WV, W) will be determined in a manner consistent with its use context, and from there, the evaluation is calculated for all services under the same category using the same metrics and parameters values, which enables us to compare these services effectively. Following this approach helped to evaluate a wide range of web services, with the possibility of comparing functionally similar services in terms of performance quality, which helped users choose the appropriate service for them from among similar services. Fig.5 displays the proposed system's flowchart diagram, which illustrates the system's processes and how data flows between them.

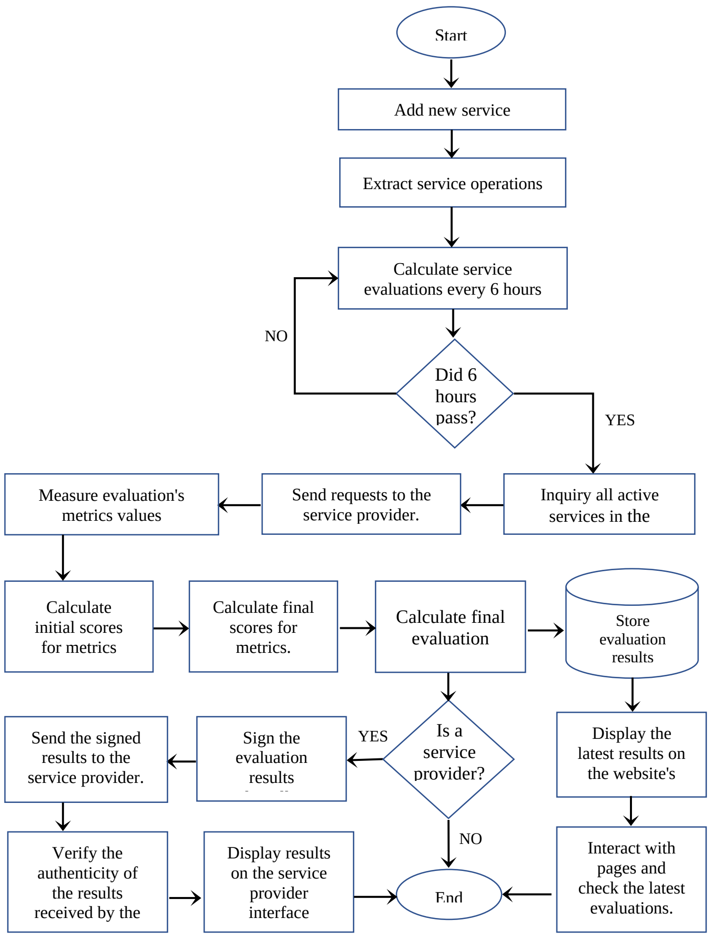
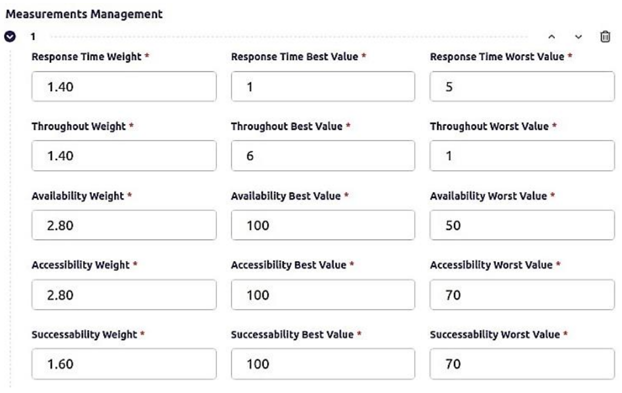

Start
Add new service
Extract service operations
Calculate service evaluations every 6 hours
NO
Did 6 hours pass?
YES
Inquiry all active services in the
Send requests to the service provider.
Measure evaluation's metrics values
Calculate initial scores for metrics
Calculate final scores for metrics.
Calculate final evaluation
Store evaluation results
Display the latest results on the website's
Is a service provider?
YES
Sign the evaluation results
Send the signed results to the service provider.
Verify the authenticity of the results received by the
Display results on the service provider interface
NO
End
Interact with pages and check the latest evaluations.

**Figure 5.** Flowchart diagram for the proposed system.

### 3.8 Presented the Proposed Mechanism

After designing the infrastructure for the evaluation mechanism, the mechanism is presented through a website so that new categories and web services can be easily added through the control panel. After adding the services, the calculating evaluation process begins dynamically, and the evaluations are displayed to users through web pages. The work is done through the following steps:

- **Adding categories:** categories are added through the control panel, where the user can add categories to services to group all services that provide similar functions and compare these services in terms of performance. As mentioned previously in this research, the weight parameter, the best value parameter, and the worst value parameter are determined through the control panel, as shown in Fig. 6, where the same values are added for all similar services to ensure effective comparison between them, as the comparison is done according to the same standards.



Measurements Management

1

| | | |
|---|---|---|
| Response Time Weight * | Response Time Best Value * | Response Time Worst Value * |
| 1.40 | 1 | 5 |
| Throughout Weight * | Throughout Best Value * | Throughout Worst Value * |
| 1.40 | 6 | 1 |
| Availability Weight * | Availability Best Value * | Availability Worst Value * |
| 2.80 | 100 | 50 |
| Accessibility Weight * | Accessibility Best Value * | Accessibility Worst Value * |
| 2.80 | 100 | 70 |
| Successability Weight * | Successability Best Value * | Successability Worst Value * |
| 1.60 | 100 | 70 |

**Figure 6.** Manage parameter values for each category through the control panel.

- **Adding web services:** Web services are added through the control panel, and work is done to evaluate the two most common types of web services: SOAP and RESTful web services. Thus, the services are added, then the service processes are extracted, along with all their details and the parameters necessary to call these processes in the correct format.
- **Calculating evaluations:** In this step, the evaluation is calculated for all web services that have been added to the website by following the evaluation mechanism that was proposed and designed. This is done through a command that inquires about all active web services on the website and executes the instructions for measuring the previously described metrics and algorithms. This command is executed periodically through cron jobs, and this is done every 6 hours daily.

  Every time a web service evaluation is calculated, the result is stored in a special record for the service. These records are used to display annual, monthly, and weekly statistics. These statistics help users understand the development of the web service's performance over the years and make a choice based on that.
- **Displaying web services and evaluation results to users:** All categories added to the site and all the services affiliated with each category are shown. These services are arranged according to the results of their evaluation while allowing users to filter services according to the metric's importance to them and their preferences. Users are provided detailed service information and evaluation results for each web service. The final evaluation result is displayed in addition to the calculated values of the metrics, as represented in Fig. 7, to achieve transparency and reliability in presenting the results.

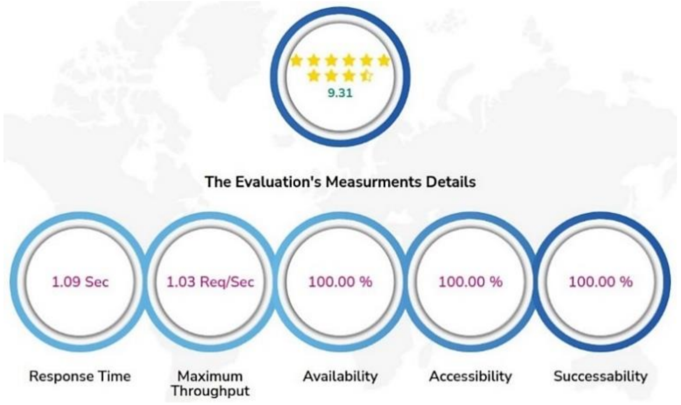


**Figure 7.** Displayed detailed evaluation results for each web service.

In addition, statistical charts are displayed for each service, showing its performance over the years, months, and weeks, as in Fig. 8, to provide users with detailed information that helps them choose the appropriate services.

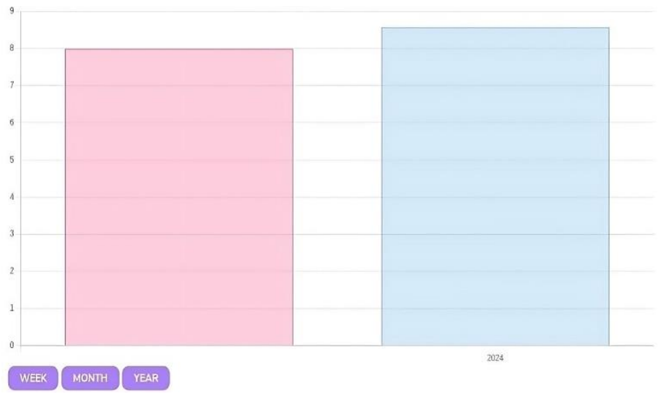


**Figure 8.** Statistical charts reflecting the service's performance.

- **Ensuring the authenticity of evaluation results:**

  In the proposed mechanism in reference [7], the evaluation results are displayed to the user only, while the proposed mechanism in this research has been designed to be general and provide the evaluation service for both web service consumers who wish to choose the appropriate web service for them from a group of functionally similar services, and web service providers who need to have their services evaluated by a trusted third party. Work has been done to ensure reliability for both types of users, where for consumers, reliability has been implicitly guaranteed by ensuring the freshness of the evaluation results and achieving transparency and objectivity by providing the user with all service information and evaluation details. As for service providers, in addition to evaluating their services and displaying them on the website, the evaluation results are sent to the service providers so that the service provider can display the evaluation result on its service description interface, and thus, all service users can know the service evaluation when using it, where this evaluation is reliable for them because it was calculated by a third party, which is the proposed evaluation mechanism, which helps them use the service with greater confidence based on the evaluation information. Fig.9 shows how a locally developed RESTful web service displays the evaluation results calculated by the proposed evaluation mechanism. Thus, consumers can learn and use service evaluation based on that knowledge. Since the evaluation results are sent from the proposed mechanism to service providers over the network, it was necessary to ensure the integrity and authenticity of the results when transmitting them over the network by signing them digitally before sending them. Two web services were developed locally to prove the ability to guarantee authenticity for different evaluated services: a SOAP web service and a RESTful web service. For SOAP web service, its evaluation results will be signed digitally from the evaluation mechanism side using the SignXML library, depending on the RSA algorithm with 2048-bit keys generated locally via OpenSSL. When the SOAP web service receives the evaluation results, it will verify the authenticity of the data by verifying the signature validity using the public key, and the evaluation result will be displayed on its description interface after that. In addition, for RESTful web services, evaluation results will be signed digitally on the evaluation mechanism side using the PyCryptodome library, depending on the RSA algorithm with the same generated keys. When the RESTful web service receives the evaluation results, it will verify the authenticity of the data by verifying the signature validity using the public key and display the evaluation result on its description interface, as represented in Fig.9.

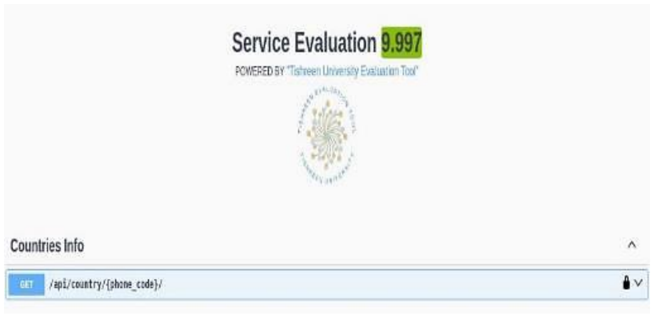

Service Evaluation 9.997
POWERED BY "Tishreen University Evaluation Tool"
Countries Info
GET
/api/country/{phone_code}/

**Figure 9.** Displayed the evaluation results on the locally designed RESTful web service description Interface.

## 4. Results and Discussion

To verify the validity of the approach followed and confirm the possibility of its implementation, many web services belonging to the various categories were added, and evaluations for these services were calculated. It has relied on one of the largest and most famous web service centers, RapidAPI, which provides more than 40 thousand web services to obtain web services. The RapidAPI team has grouped functionally similar services, forming more than 500 groups. For example, one group contains services that belong to flight data, and another one contains services related to sending emails and validating email addresses [9], [24]. Services were grouped into categories to compare similar functionality services each other, where for each category of services, work was done depending on the following steps:

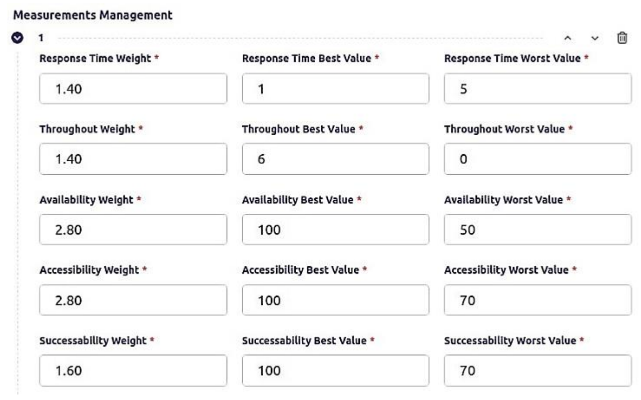
- Determining the parameters' values: As discussed before, for all services that belong to the same category, the same parameters' values will be used to evaluate services based on the same criteria so we can compare them effectively.
- Measuring the values of the five metrics for all web services under this category.
- Calculating the initial score of services' metrics depending on their measured values in step 2 and the best value and worst value parameters specified for the category in step 1.
- Inferring the final score for services' metrics deepening on the initial score calculated in step 2 and the weight parameter determined for the category in step 1.
- Calculating the final evaluation for each web service by summing the values of its metrics final scores cumulatively.

This research will introduce three test cases for three web services categories.

**- First test case:** Three services that provide translation functionality are manually selected. These services are Text Translator[7], Google Translate[8], and Tribal Mail[9]. This is according to the information provided by the RapidAPI pages for these services. Data and results for each step are summarized by the following:

- Some translation web services may use advanced natural language processing (NLP) techniques or handle complex language process functionality. In its role, this can impact response time for these services, mainly when they handle long texts and some complex tasks. However, it's always important for these services to be consistently available and handle requests successfully without errors. Based on that, in this test case, high weights were assigned for availability, accessibility, and successability, and the threshold of their worst accepted values was increased. In contrast, the weights for response time and maximum throughput were given lower values than previous metrics, and the threshold of their worst value decreased. The values of all the metrics' parameters are represented in the Fig.10.

[7] *Text Translator*
[8] *Google Translate*
[9] *Tribal Mail – Translate*


Measurements Management

1

| | | |
|---|---|---|
| Response Time Weight * | Response Time Best Value * | Response Time Worst Value * |
| 1.40 | 1 | 5 |
| Throughout Weight * | Throughout Best Value * | Throughout Worst Value * |
| 1.40 | 6 | 0 |
| Availability Weight * | Availability Best Value * | Availability Worst Value * |
| 2.80 | 100 | 50 |
| Accessibility Weight * | Accessibility Best Value * | Accessibility Worst Value * |
| 2.80 | 100 | 70 |
| Successability Weight * | Successability Best Value * | Successability Worst Value * |
| 1.60 | 100 | 70 |

**Figure 10.** The values of metrics' parameters for translation services.

- Table 2 shows the measured values for all translation services' metrics.

**Table 2.** Measured values for translation services' metrics.

| Service | RT | TH | AV | ACC | AVV |
|---|---|---|---|---|---|
| Text Translator | 1.092 | 1.025 | 100 | 100 | 100 |
| Google Translate | 1.266 | 0.829 | 100 | 100 | 100 |
| Tribal Mail | 4.190 | 0.542 | 100 | 100 | 100 |

- The metrics' initial scores will be calculated depending on measured values and parameters. Table 3 displays the metrics' initial scores.

**Table 3.** Initial scores for translation services metrics.

| Service | RT | TH | AV | ACC | AVV |
|---|---|---|---|---|---|
| Text Translator | 0.976 | 0.170 | 1 | 1 | 1 |
| Google Translate | 0.933 | 0.138 | 1 | 1 | 1 |
| Tribal Mail | 0.202 | 0.090 | 1 | 1 | 1 |

- Table 4 displays the final scores for all translation services' metrics.

**Table 4.** Final scores for translation services metrics.

| Service | RT | TH | AV | ACC | AVV |
|---|---|---|---|---|---|
| Text Translator | 1.367 | 0.239 | 2.80 | 2.80 | 1.60 |
| Google Translate | 1.306 | 0.193 | 2.80 | 2.80 | 1.60 |
| Tribal Mail | 0.283 | 0.126 | 2.80 | 2.80 | 1.60 |

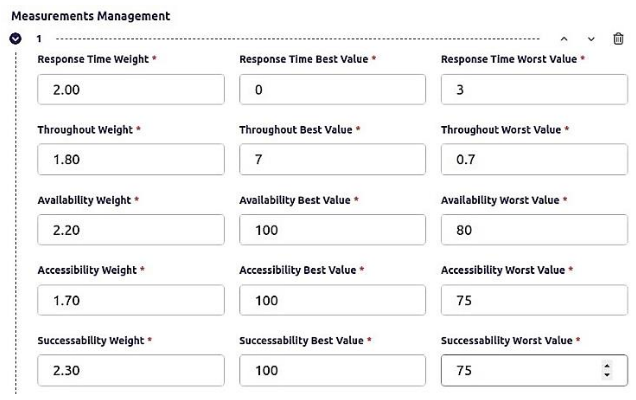
- Final translation services evaluations will be displayed in a decreased order based on the evaluation results in Table 5.

**Table 5.** Translation services evaluation results.

| Service | Service Evaluation |
|---|---|
| Text Translator | 8.806 |
| Google Translate | 8.699 |
| Tribal Mail | 7.6097 |

**- Second test case:** Three services were chosen manually, and they provide the function of creating temporary email addresses or verifying the validity of addresses. These services are Temp Mail[10], Email Verification[11], and Email Validator API[12], according to the information provided by the RapidAPI pages for these services. Data and results for each step are summarized by the following:

- For email web services, the main tasks achieved by these services are email verifications or generating temporary emails. Because these functionalities are simple, the response time for these services is expected to be small. In addition, these services need to provide stable functionalities, so they should be available and handle requests successfully most of the time. For this reason, in this test case, the weights of response time and maximum throughput were increased, in addition to maximizing the threshold of the worst accepted value for them and increasing the limit of their expected best value. While keeping the weights of availability, accessibility, and successability high because they are important too, their weights were decreased to increase the weights of response time and maximum throughput. The values of all the metrics' parameters are represented in the Fig.11.


Measurements Management

1

| | | |
|---|---|---|
| Response Time Weight * | Response Time Best Value * | Response Time Worst Value * |
| 2.00 | 0 | 3 |
| Throughout Weight * | Throughout Best Value * | Throughout Worst Value * |
| 1.80 | 7 | 0.7 |
| Availability Weight * | Availability Best Value * | Availability Worst Value * |
| 2.20 | 100 | 80 |
| Accessibility Weight * | Accessibility Best Value * | Accessibility Worst Value * |
| 1.70 | 100 | 75 |
| Successability Weight * | Successability Best Value * | Successability Worst Value * |
| 2.30 | 100 | 75 |

**Figure 11.** The values of metrics' parameters for email services.

[10] *Temp Mail*
[11] *Email Verification*
[12] *Email Validator API*

- Table 6 shows the measured values for all email services' metrics.

**Table 6.** Measured values for email services' metrics.

| Service | RT | TH | AV | ACC | AVV |
|---|---|---|---|---|---|
| Temp Mail | 1.035 | 0.971 | 100 | 100 | 100 |
| Email Verification | 1.694 | 0.754 | 100 | 100 | 100 |
| Email Validator | 11.60 | 0.087 | 100 | 100 | 100 |

- The metrics' initial scores will be calculated depending on measured values and parameters. Table 7 displays the metrics' initial scores.

**Table 7.** Initial scores for email services metrics.

| Service | RT | TH | AV | ACC | AVV |
|---|---|---|---|---|---|
| Temp Mail | 0.654 | 0.043 | 1 | 1 | 1 |
| Email Verification | 0.435 | 0.007 | 1 | 1 | 1 |
| Email Validator | 0 | 0 | 1 | 1 | 1 |

- Table 8 displays the final score results for all email services' metrics.

**Table 8.** Final scores for email services metrics.

| Service | RT | TH | AV | ACC | AVV |
|---|---|---|---|---|---|
| Temp Mail | 1.31 | 0.077 | 2.20 | 1.70 | 2.30 |
| Email Verification | 0.87 | 0.012 | 2.20 | 1.70 | 2.30 |
| Email Validator | 0 | 0 | 2.20 | 1.70 | 2.30 |

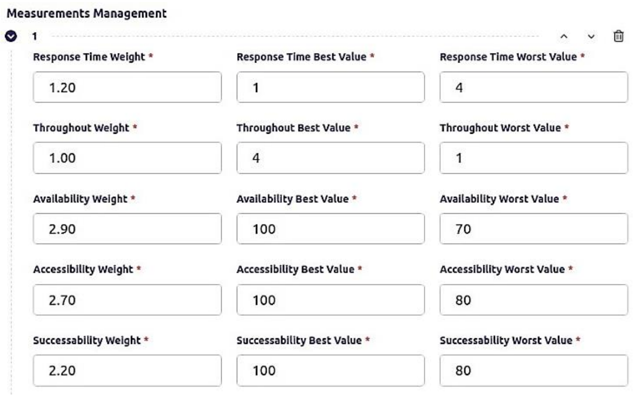
- Final email services evaluations will be displayed in decreased order, based on evaluation results in Table 9.

**Table 9.** Email services evaluation results.

| Service | Evaluation Result |
|---|---|
| Temp Mail | 7.587 |
| Email Verification | 7.082 |
| Email Validator | 6.2 |

**- Third test case:** Manually select three services that provide users with the latest news related to various fields. These services are News API[13], Indonesian News Feed[14], and Arabic news API[15], according to the information provided by the RapidAPI pages for these services. Data and results for each step are summarized by the following:

- News web services will work to keep users updated with the latest news; querying large numbers of news and sending substantial payloads over the network can noticeably affect response times, sometimes by several seconds. While these services provide straightforward functionalities, they must handle requests efficiently and work correctly most of the time. Otherwise, their quality will be perceived as low. Therefore, the weights of response time and maximum throughput have been decreased compared to translation and email services; in addition, the threshold of the expected worst value for these parameters has been decreased, and the limit of the expected best value limit for them has been increased. In addition, the weights of other parameters have been increased, and the thresholds of their worst allowed value have been raised to ensure that user requests are handled correctly at all times. The values of all the metrics' parameters are represented in the Fig.12.


Measurements Management

1

| | | |
|---|---|---|
| Response Time Weight * | Response Time Best Value * | Response Time Worst Value * |
| 1.20 | 1 | 4 |
| Throughout Weight * | Throughout Best Value * | Throughout Worst Value * |
| 1.00 | 4 | 1 |
| Availability Weight * | Availability Best Value * | Availability Worst Value * |
| 2.90 | 100 | 70 |
| Accessibility Weight * | Accessibility Best Value * | Accessibility Worst Value * |
| 2.70 | 100 | 80 |
| Successability Weight * | Successability Best Value * | Successability Worst Value * |
| 2.20 | 100 | 80 |

**Figure 12.** The values of metrics' parameters for news services.

- Table 10 shows the measured values for all news services' metrics.

**Table 10.** Measured values for news services metrics.

| Service | RT | TH | AV | ACC | AVV |
|---|---|---|---|---|---|
| News API | 1.199 | 1.036 | 100 | 100 | 100 |
| Indonesian News | 2.362 | 0.740 | 100 | 100 | 100 |
| Arabic news | 6.440 | 0.263 | 100 | 100 | 100 |

- The metrics' initial scores will be calculated depending on measured values and parameters. Table 11 displays the metrics' initial scores.

[13] *News API*
[14] *Indonesian News Feed*
[15] *Arabic news API*

**Table 11.** Initial scores for news services metrics.

| Service | RT | TH | AV | ACC | AVV |
|---|---|---|---|---|---|
| News API | 0.933 | 0.012 | 1 | 1 | 1 |
| Indonesian News | 0.545 | 0 | 1 | 1 | 1 |
| Arabic news | 0 | 0 | 1 | 1 | 1 |

- Table 12 displays the final score results for all news services' metrics.

**Table 12.** Final scores for news services metrics.

| Service | RT | TH | AV | ACC | AVV |
|---|---|---|---|---|---|
| News API | 1.119 | 0. 012 | 2.90 | 2.70 | 2.20 |
| Indonesian News | 0.654 | 0 | 2.90 | 2.70 | 2.20 |
| Arabic news | 0 | 0 | 2.90 | 2.70 | 2.20 |

- Final email services evaluations will be displayed in decreased order, deepening on evaluation results in Table 13.

**Table 13.** News services evaluation results.

| Service | Evaluation Result |
|---|---|
| News API | 8.931 |
| Indonesian News | 8.454 |
| Arabic news | 7.8 |

## 5. Work Limitation:

**-** The parameter values of the metrics in various test cases were assigned based on specific assumptions clarified previously for each case. However, these values may be adjusted based on real-life experiments or the development of new techniques that enhance the speed of certain services, such as translation services. Consequently, the mechanism has been designed to manage metrics via a control panel, as illustrated in Fig. 10, Fig. 11, and Fig. 12, allowing for easy adjustments to accommodate different conditions.

- The proposed evaluation mechanism will be a third-party service located on a separate server and provided to different types of users through a website. Users can view the various categories and web services evaluated and interact with the evaluation mechanism through the website pages. The evaluation results shown in the tables are not affected by the location and environment of the users, as these evaluations are not calculated on the user's side but on a separate server. Therefore, the results will be the same for all users regardless of location and environment. However, the evaluation results may differ if the location or environment of the proposed mechanism's server changes due to the difference in some factors, such as internet speed, and the impact of these factors on the values of some metrics, such as response time when measured. However, despite this difference, the results of the comparison between the quality of functionally similar web services are not affected because the values of the standards are measured. The quality of service is calculated for all services evaluated within the exact location and environment, ensuring the effectiveness and integrity of the comparison between functionally similar services.

## 6. Conclusion

This research proposes a mechanism for evaluating web services. The mechanism follows the evaluation strategy based on QoS metrics, where five metrics are chosen to be measurable from the user side. We measured these metrics and calculated evaluations for the two most common types of web services: SOAP web services and REST web services. The mechanism is presented through a website with the ability to add unlimited numbers of web services and the ability to group similar services to help users compare similar services and filter services according to the importance of the metric to them, in addition to calculating ratings periodically, to ensure the freshness and the reliability of the evaluations, in addition to display statistics to users to provide them with information about the development of various web services over the years and choosing the web service in light of that. In the future, by taking advantage of the proposed mechanism, we seek to design an evaluation mechanism that covers multiple evaluation strategies. For example, designing a mechanism that calculates the popularity of the service for users in addition to the quality of service metrics (QoS) in an attempt to study the integration of different strategies with each other, as well as its effect in giving more objective and comprehensive evaluations of web services.

**Acknowledgments-**

The authors would like to reveal their appreciation and gratitude to the respected reviewers and editors for their constructive comments.

**Conflict of interest**

The authors declare that there are no conflicts of interest regarding the publication of this manuscript.

**Author Contribution Statement**

Aya Joumaa.: developed the theory and performed the computations.

Basel Hasan and Nasser Nasser.: proposed the research problem, verified the analytical methods, and supervised the findings of this work.

All authors discussed the results and contributed to the final manuscript.

## References

[1] A. V. Tokmak, A. Akbulut, and C. Catal, “Web service discovery: Rationale, challenges, and solution directions,” *Comput Stand Interfaces*, vol. 88, p. 103794, 2024, doi: https://doi.org/10.1016/j.csi.2023.103794.

[2] A. Soni and V. Ranga, “API Features Individualizing of Web Services: REST and SOAP,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 9S, pp. 664–671, Aug. 2019, doi: https://doi.org/10.35940/ijitee.I1107.0789S19.

[3] K. Zatwarnicki, “Providing Predictable Quality of Service in a Cloud-Based Web System,” *Applied Sciences*, vol. 11, no. 7, p. 2896, Mar. 2021, doi: https://doi.org/10.3390/app11072896.

[4] L. Purohit and S. Kumar, “Web Services in the Internet of Things and Smart Cities: A Case Study on Classification Techniques,” *IEEE Consumer Electronics Magazine*, vol. 8, no. 2, pp. 39–43, Mar. 2019, doi: https://doi.org/10.1109/MCE.2018.2880808.

[5] O. V. Polska, R. K. Kudermetov, and V. V. Shkarupylo, “An Approach Web Service Selection By Quality Criteria Based On Sensitivity Analysis of MCDM Methods,” *Radio Electronics, Computer Science, Control*, no. 2, pp. 133–143, Jul. 2021, doi: https://doi.org/10.15588/1607-3274-2021-2-14.

[6] H. Zhang, Z. Shao, H. Zheng, and J. Zhai, “Web Service Reputation Evaluation Based on QoS Measurement,” *The Scientific World Journal*, vol. 2014, pp. 1–7, 2014, doi: https://doi.org/10.1155/2014/373902.

[7] T. Bouasker, M. Langar, and R. Robbana, “QoS monitor as a service,” *Software Quality Journal*, vol. 28, no. 3, pp. 1279–1301, Sep. 2020, doi: https://doi.org/10.1007/s11219-020-09514-1.

[8] S. Romdhani, G. Vargas-Solar, N. Bennani, and C. Ghedira-Guegan, “QoS-based Trust Evaluation for Data Services as a Black Box,” in *2021 IEEE International Conference on Web Services (ICWS)*, Chicago, IL, USA, pp. 476–481, 2021, doi: https://doi.org/10.1109/ICWS53863.2021.00067.

[9] Z. Song, O. Rowader, Z. Li, M. Tello, and E. Tilevich, “Quality of Information Matters: Recommending Web Services for Performance and Utility,” in *2022 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Bangkok, Thailand, pp. 41–48. 2022, doi: https://doi.org/10.1109/CloudCom55334.2022.00016.

[10] C. Li, B. Cheng, J. Chen, P. Gu, N. Deng, and D. Li, “A Web Service Performance Evaluation Approach Based on Users Experience,” in *2011 IEEE International Conference on Web Services*, Washington, DC, USA, pp. 734–735. 2011, doi: https://doi.org/10.1109/ICWS.2011.29.

[11] S. Wu et al., “Popularity-Aware and Diverse Web APIs Recommendation Based on Correlation Graph,” *IEEE Trans Comput Soc Syst*, vol. 10, no. 2, pp. 771–782, Apr. 2023, doi: https://doi.org/10.1109/TCSS.2022.3168595.

[12] A. Sunyaev, “Web Services,” in Internet Computing, Cham*: Springer International Publishing* pp. 155–194, 2020, doi: https://doi.org/10.1007/978-3-030-34957-8_6.

[13] K. Kumar, A. K. Jain, R. G. Tiwari, N. Jain, V. Gautam, and N. K. Trivedi, “Analysis of API Architecture: A Detailed Report,” in *2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT)*, Bhopal, India, Apr. 2023, pp. 880–884. doi: https://doi.org/10.1109/CSNT57126.2023.10134658.

[14] J. Tihomirovs and J. Grabis, “Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics,” *Information Technology and Management Science*, vol. 19, no. 1, Jan. 2016, doi: https://doi.org/10.1515/itms-2016-0017.

[15] M. A. Fdheel, I. K. Abboud, and A. S. Hassan, “Web Services Design and Implementation through C# .NET,” *J. eng. Sustain. Dev.*, vol. 18, no. 4, pp. 141–154, Jul. 2014. [Online]. Available: https://jeasd.uomustansiriyah.edu.iq/index.php/jeasd/article/view/844

[16] J. Juneau and T. Telang, “RESTful Web Services,” *in Java EE to Jakarta EE 10 Recipes*, Berkeley, CA: Apress, pp. 511–530. 2022,doi: https://doi.org/10.1007/978-1-4842-8079-9_13.

[17] S. U. Meshram, "Evolution of Modern Web Services – REST API with its Architecture and Design," *IJRESM*, vol. 4, no. 7, pp. 83–86, Jul. 2021. [Online]. Available: https://journal.ijresm.com/index.php/ijresm/article/view/970

[18] B. Barhoum, *Information systems security*, Latakia, Syria: Directorate of Books and Publications - Tishreen University, 2020.

[19] S. M. Suhael, Z. A. Ahmed, and A. J. Hussain, “Proposed Hybrid Cryptosystems Based on Modifications of Playfair Cipher and RSA Cryptosystem,” *Baghdad Sci. J*, vol. 21, no. 1, pp. 151-160, Jan. 2024. doi: https://doi.org/10.21123/bsj.2023.8361

[20] S. G. Chaloop and M. Z. Abdullah, “Enhancing Hybrid Security Approach Using AES and RSA Algorithms,” *J. eng. Sustain. Dev.*, vol. 25, no. 4, pp. 58–66, Jul. 2021, doi: https://doi.org/10.31272/jeasd.25.4.6.

[21] K. K. Jabbar, F. Ghozzi, and A. Fakhfakh, “Robust Color Image Encryption Scheme Based on RSA via DCT by Using an Advanced Logic Design Approach,” *Baghdad Sci. J*, vol. 20, no. 6(Suppl.), p. 2593-2607, Dec. 2023, doi: https://doi.org/10.21123/bsj.2023.8715.

[22] J. Miliauskaite, “Quality of service: Concept analysis,” *CEUR Workshop Proc*, vol. 924, pp. 235–240, Jan. 2012. [Online]. Available: https://ceur-ws.org/Vol-924/paper24.pdf.

[23] Y. Wang and J. Vassileva, “A Review on Trust and Reputation for Web Service Selection,” in *27th International Conference on Distributed Computing Systems Workshops (*ICDCSW’07), Toronto, ON, Canada pp. 25–25., 2007, doi: https://doi.org/10.1109/ICDCSW.2007.16.

[24] J. C. Alonso, A. Martin-Lopez, S. Segura, J. M. Garcia, and A. Ruiz-Cortes, “ARTE: Automated Generation of Realistic Test Inputs for Web APIs,” *IEEE Trans Softw Eng*, vol. 49, no. 1, pp. 348–363, Jan. 2023, doi: https://doi.org/10.1109/TSE.2022.3150618.