

**Humanities and Educational
Sciences Journal**

ISSN: 2617-5908 (print)



**مجلة العلوم التربوية
والدراسات الإنسانية**

ISSN: 2709-0302 (online)

Enhancing E-learning Applications based on Modified Web Service Model^(*)

<https://doi.org/10.55074/hesj.vi56.1948>

Adeeb Mohammed Hider Alqershi

PhD Researcher

E-D Learning Faculty, Taiz University, Yemen

Adeeb.alqershi@taiz.edu.ye

Gamal Ahmed Ahmed Abdulla Alawi

Prof

E-D Learning Faculty, Taiz University, Yemen

Jamal.alawi@taiz.edu.ye

^(*) Received: 11/3/2026

Accepted: 21/4/2026

^(*) Journal Website:

<http://hesj.org/ojs/index.php/hesj/index>

تحسين تطبيقات التعليم عن بعد باستخدام نموذج خدمات ويب معدل

أديب محمد حيدر القرشي

باحث بكلية التعليم عن بعد - جامعة تعز - اليمن

جمال أحمد أحمد عبدالله علوي

أستاذ تكنولوجيا التعليم بكلية التعليم عن بعد - جامعة تعز - اليمن

الملخص

نبغي أن تكون خدمات التعلّم الإلكتروني متاحة للمستخدم بكفاءة. ويستغرق الحصول على خدمات الويب من الخادم الأصلي وقتًا أطول، وذلك اعتمادًا على سرعة الإنترنت، وحركة مرور الشبكة، ومشكلات الشبكات، ومدى توفر الخادم. لذلك، تبرز الحاجة إلى توفير خدمات الويب بشكل مستمر كلما طلبها المستخدم. في هذه الورقة، تم اقتراح نموذج خدمات الويب للتعلّم الإلكتروني والذي يتضمن مكونات جديدة وفعالة تساهم في تحسين أداء تطبيقات التعلّم الإلكتروني، مثل زمن الاستجابة، واستخدام وحدة المعالجة المركزية، والمزامنة. بالإضافة إلى ذلك، يعمل نموذج ELWSM على تحسين إتاحة خدمات التعلّم الإلكتروني وموثوقيتها واستقرارها. وقد أدى تطبيق النموذج إلى خفض التكلفة وزمن الاستجابة بنسبة 47.1%. علاوة على ذلك، أسهم نموذج ELWSM في تقليل استخدام وحدة المعالجة المركزية بنسبة 61.3% كما عمل مكوّن معزّز استجابة واجهة المستخدم الرسومية على تحسين تفاعل المستخدم مع النموذج بنسبة 99%. وبناءً على ذلك، تؤكد هذه النتائج قدرة النموذج المقترح على تحقيق الاستقرار، والمرونة، والاتساق، والإتاحة، والموثوقية في خدمات التعلّم الإلكتروني. الكلمات المفتاحية: التعلّم الإلكتروني، خدمات الويب، نموذج خدمات الويب للتعلّم الإلكتروني (ELWSM)، زمن الاستجابة، استخدام وحدة المعالجة المركزية، معزّز استجابة واجهة المستخدم الرسومية.

Enhancing E-learning Applications based on Modified Web Service Model

Adeeb Mohammed Hider Alqershi

PhD Researcher

E-D Learning Faculty, Taiz University, Yemen

Gamal Ahmed Ahmed Abdulla Alawi

Prof

E-D Learning Faculty, Taiz University, Yemen

Abstract

E-learning services must be available in consumer hand efficiently. Acquiring Web Services (WS) from original server takes a more time based on Internet speed, Internet traffic, networks troubleshooting and server availability. Thus, there is needing to set WS available each time that user request.

In this paper, a proposed E-learning Web Services Model (ELWSM) is conducting with new efficient components that are supporting E-learning application performance such as Response Time (RT), CPU Utilization and Synchronization.

In addition, ELWSM enhance e-learning services availability, reliability and Stability. Applying ELWSM reduced the cost and response time by 47.1%.

Furthermore, ELWSM save CPU Utilization by 61.3%. Also, ELWSM by Graphical User Interface Response Enhancer (GUIRE) enhances the model interaction by 99%.

Consequently, these results ensure stability, resilience, consistency, availability, and reliability.

Keywords: E-learning, WS, ELWSM, RT, CPU utilization, GUIRE

Introduction:

In recent years, e-learning has grown rapidly and become an essential part of modern education. At the same time, e-learning platforms suffer from many issues related to data volume, interactivity, accessibility, and reliability. To overcome these limitations, web services play an important role in modern system development

Literature Review:

A *web service* is a software system designed to support interoperable machine-to-machine interaction over a network" (Ali et al., 2021). WS is a model that allow organization to publish their services online (AltexSoft, 2025). There are two models for implement WS, which are SOAP and REST (Pautasso et al., 2021).

1. WS Traditional Model:

As shown in Figure (1), the WS standard model has three components: service provider, service registry and consumer. Practically, service provider creates his own service(s), then the services(s) must be published on registry hosting, which hav additional information about provider such as address and port type. After that the service will be available for consumer (Dustdar et al., 2022).

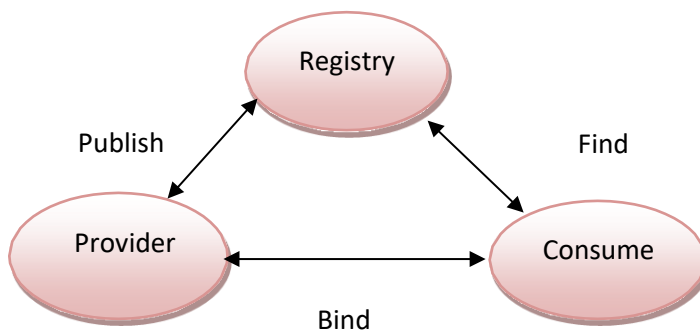


Figure (1): WS model entities (Dustdar et al., 2022)

2. Web caching Techniques

Several studies have demonstrated that caching strategies significantly enhance web service performance by reducing latency and minimizing redundant service invocations (Zhang et al., 2023).

3. HTTP Cache Partitioning (Double-Keyed Caching):

This technique addresses the privacy flaws of a "global" cache by segregating stored resources based on the top-level site domain. This prevents cross-site tracking where a malicious site detects a user's history by checking resource load times.

Advantage: Provides a robust efficiency against "cache-probing" and "privacy-leaking" side-channel attacks (Osmani, 2025).

Disadvantage: Negatively impacts performance for shared resources (like CDNs), leading to higher bandwidth consumption and slightly longer load times (Goethem et al., 2020).

4. AI and Machine Learning-Based Caching:

Discussion: Modern caching frameworks now utilize Reinforcement Learning (RL) to predict which web objects a user will likely request next, moving beyond static "Least Recently Used" (LRU) models.

Advantage: Significantly optimizes "hit rates" by adapting to real-time user behavior patterns (Ali et al., 2020).

Disadvantage: Requires substantial computational resources (CPU/RAM) to run prediction models, which can be taxing for mobile clients (Wibowo et al., 2025).

5. Progressive Edge Caching (MEC-Integrated):

Discussion: This technique leverages Multi-access Edge Computing (MEC) to cache web content at the network's edge, effectively bringing the "client cache" experience to the router.

Advantage: Drastically reduces "round-trip time" (RTT) and minimizes the load on origin servers (Kommareddy, 2025).

Disadvantage: Introduces complexity in maintaining "cache consistency," as data must be synchronized across numerous distributed edge nodes (Mansuri, 2025).

6. Cache-Based Computational Reuse

Discussion: This involves caching the "execution state" or rendered output of JavaScript components rather than just the raw source code.

Advantage: Conserves battery life and reduces processor heat on client devices by preventing redundant script execution (Wibowo et al., 2025).

Disadvantage: Expands the "attack surface" for side-channel attacks, as an attacker might extract sensitive data from the cached computational state (Goethem et al., 2020).

7. Service-Oriented E-Learning Architecture

Several studies emphasize the importance of Service-Oriented Architecture (SOA) in e-learning. Mohammed (2008) demonstrated that using web services allows e-learning systems to:

Separate learning functions into independent services

7.1 Reuse educational components

7.2 Integrate third-party learning tools easily

This approach reduces system complexity and improves maintainability.

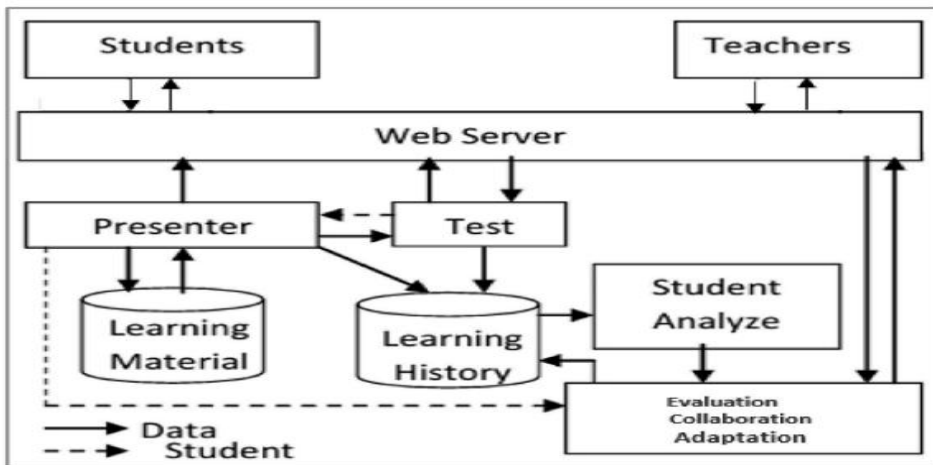


Figure (2): E-learning system design architecture (Balla, 2013)

8. RESTful Web Services in Modern Platforms

Recent research has shifted toward RESTful web services due to their simplicity and performance.

Mostefai et al. (2025) proposed a user-centric RESTful architecture for e-learning platforms. Their results showed measurable improvements in:

8.1 System usability.

8.2 User satisfaction.

8.3 Interaction efficiency.

This confirms that web services are not only technical enablers but also directly affect learning experience.

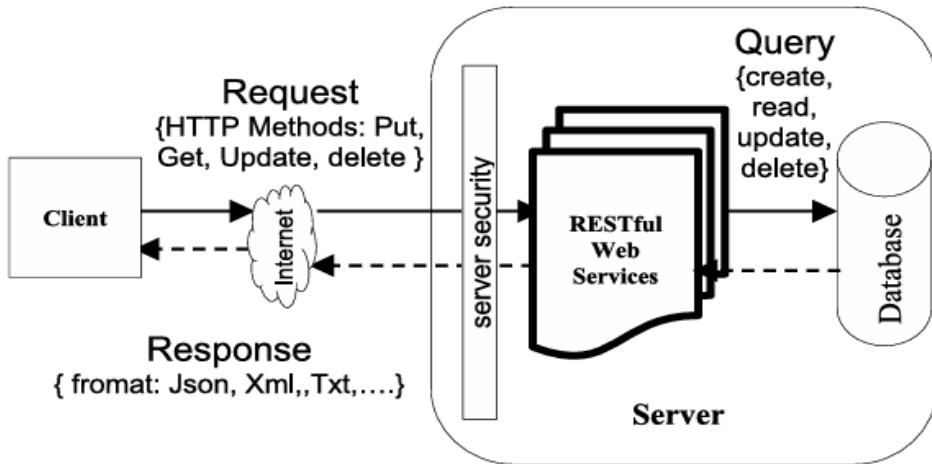


Figure (3): RESTful web services architecture (Waller et al., 2013).

Theoretical framework:

In this section, a detailed description of new proposed model of Web Services is demonstrated. Web services play a vital role in distributed e-learning platforms by enabling seamless service composition and efficient resource sharing among heterogeneous systems (Halim et al., 2020).

Actually, this includes some extensions and modifications to the traditional Web Services Model. The suggested model has been developed by supplement of many effective components. The purpose of the suggested components is to improve the performance of WS. Consequentially, the proposed model allows more efficient management for request and response of Web Services within organizations, as well as for Web Services providers.

To evaluate the performance of the proposed model, essential measurement metrics were employed. These metrics include: Response Time and CPU Utilization. These factors will clarify the degree of performance enhancement of the targeted organizations.

1. Proposed WS new components

The new enhancement WS proposed model as in figure (4) has additional

effective components that have been added into client side WS as listed below:

1. Graphical User Interface Enhancer
2. Client Cache Manager (CCM)
3. Delivery Checker (DC)
4. Local Client Cache (LCC)
5. Evictor

The figure bellow demonstrates t proposed modified WS model

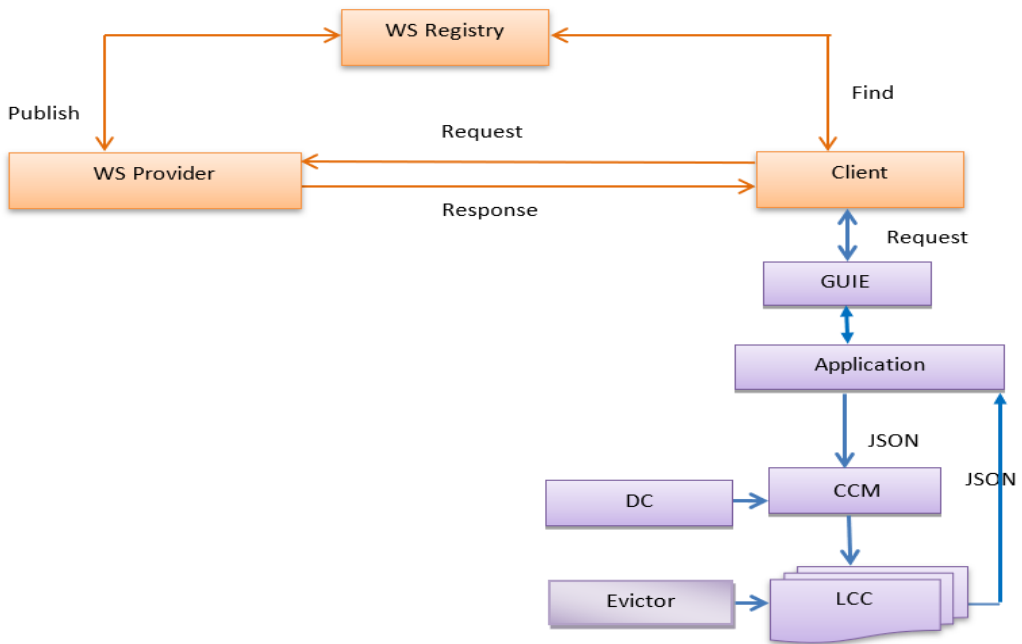


Figure (4): The proposed modified WS modified Model

1. *Client Cache Manager (CCM):*

This component has developed to store JSON files that have been sent into client side.

These files will be stored permanently. Then, any new data will be concerned modifications on those files.

1.1 *CCM Functionality:*

CCM has been designed to achieve many of processes, which are demonstrated bellow.

- 1) Create Local Client cache LCC
- 2) Receive updated data from server as JSON file.
- 3) Update Local Client Cache (*LCC*) by assign delivered JSON file from WS Provider by perform these steps
 1. Move data from delivered JSON file into LCC
 2. When step (1) done successfully, the delivered JSON file will be deleted to save memory space. The ELMWS has only one LCC, so saving memory space is essential factor.
- 4) Perform a search operation on LCC and return data.
- 5) Redirect the search process to WS server, when the LCC doesn't match the searched data.

1.2 CCM Contribution in ELMWS:

- 1- Reduce dependency on the server by perform all process in client side, thereby network bottlenecks and server load will be reduced automatically
- 2- Access Web Services offline as possible.
- 3- There is only one updated (LCC), which saves memory space.
- 4- Reduce response time.

1.3 CCM Algorithms

Algorithm: *Update Local Cache*

Input Data: *notifyFilePath , LCCFilePath*

```

1  begin
2    deliveredFile=loadJsonFile(notifyFilePath)
3    LCC= loadJsonFile(LCCFilePath)
4    File[] = deliveredFile.getJson(operationType)
/* search for operation node which is insert, update or delete*/
5    for each node in nodes do
6      if file[1].value= 1 then /* 1 = insert operation */
7        insertIntoLCC()
8      else if file[1].value= 2 then /* 2 = update operation */
9        updateLCC()
10     else
11       deleteFromLCC()
12     end for
13     saveFile(LCC)
14 end

```

This algorithm shows how Local Client Cache (LCC) is updated from delivered notification file.

First, Loading both LCC and delivered notifyFile into deliveredFile and LCC objects.

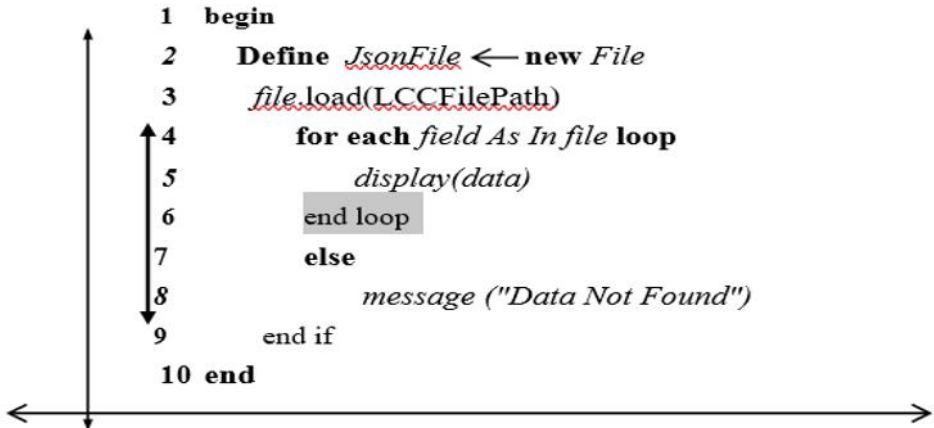
Second, get JSON node that holds type of notification operation, which is insert, update or delete operation.

Third, loop during all notifying file, if node value equal **1** then **insertIntoLCC** method will invoked to insert record(s), or if the node value will be **2** then **updateLCC** method will invoked to update LCC, else the operation will be deleted by invoking **deleteFromLCC** method. Finally, **saveJSONFile** method saves all previous modifications into LCC.

Algorithm: Search in Local Cache

Input Data: queryString, LCCFilePath

Result: get record from LC



This pseudo code shows searching process steps as:

1. define object for JSON file.
2. loading Json file from Local Client cache into *file* object by **load** method which accepts LCCFilePath then, for each loop works to fetch interested data, otherwise error message is raised.
3. Display method used to display fetched data.

2. Delivery Checker (DC)

Delivery Checker: is Modified WS Model component within the Client Side that has been developed to check if JSON have been delivered successfully without losing from WS Provider to client. The DC check either a JSON file is new or updated file.

2.1 DC Functionality:

This component ensures that the consumer gets all interested notifications without loss by achieving these steps

- 1- Get notify from server about ready of JSON files (either updated or new).
- 2- Receive delivered data.
- 3- Check if the JSON file doesn't lose any data by comparing count of received records with count of records in Server Meta Cache.
- 4- Check if notifications file has been reflected correctly on the local cache.
- 5- Check if 3, 4 steps done successfully, then send notify to evictor in Server side to evict those notifications, otherwise, evictor doesn't evict the data from MC and data within MC will be available any time.

Figure (5) shows DC functionality among modified WS model components

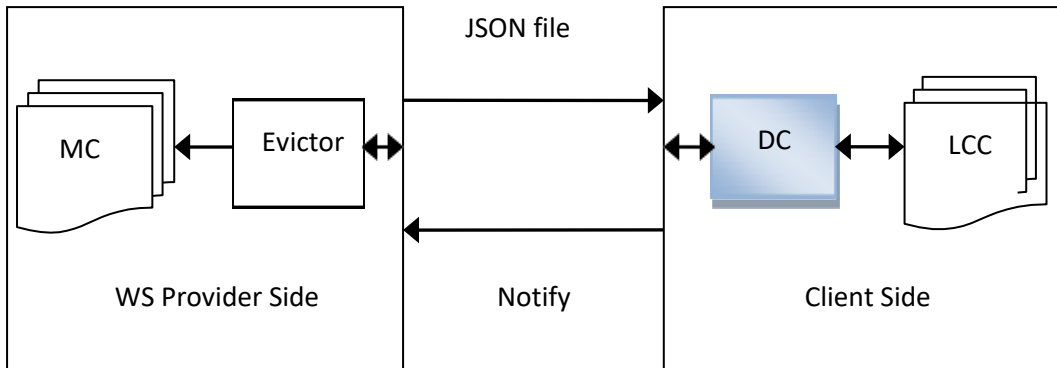


Figure (5): DC functionality in modified WS Model

2.2 DC Contribution

DC contributes substantially to the enhancement of ELMWS by avoiding many scenarios losing data (JSON Files) such as:

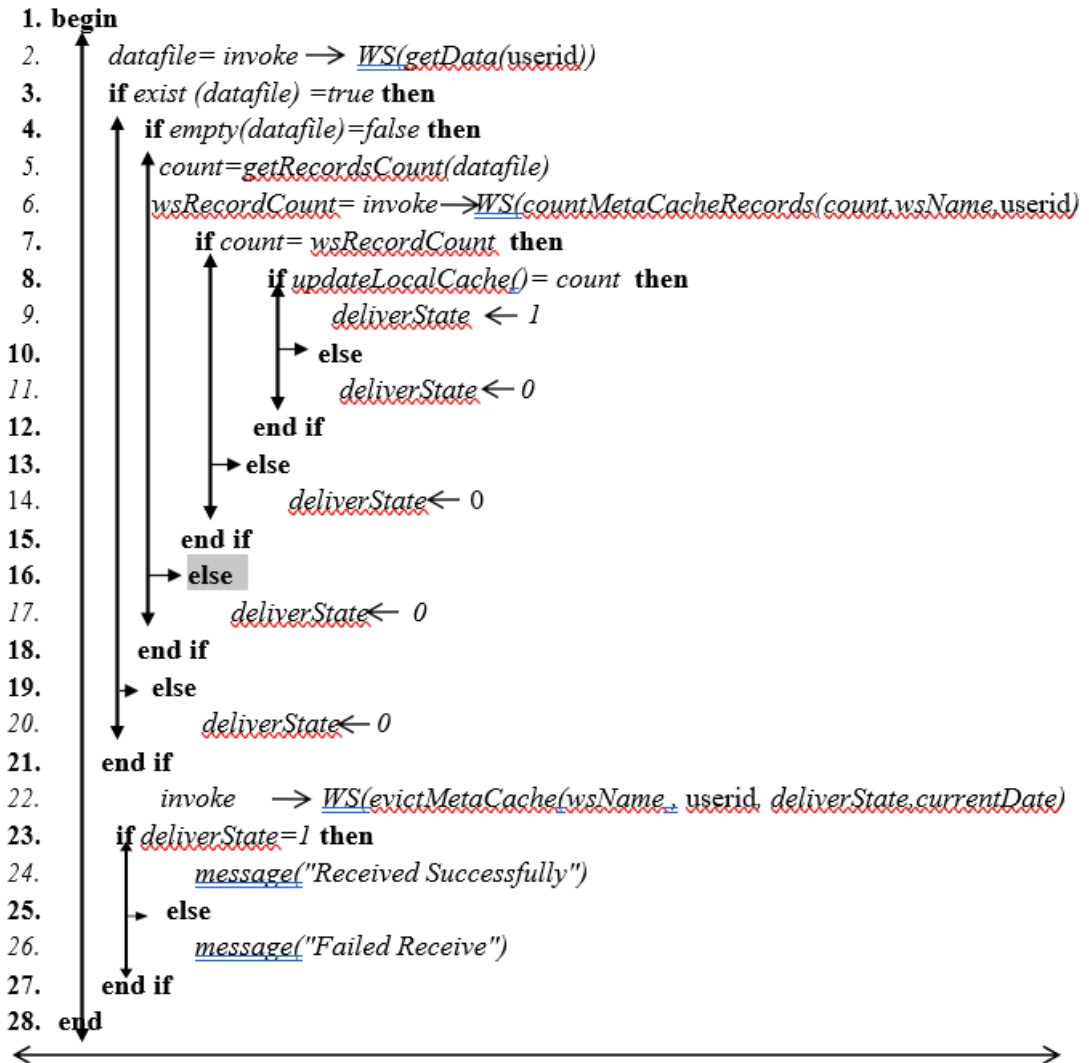
- 1- Loss delivered data due to internet or network interruption.
- 2- Loss delivered data due to unknown storage destination.
- 3- Loss delivered data due to limited disk space.

2.3 DC Algorithm

Algorithm: *Delivery Checker*

Input Data: userid, datafile, count, wsName, deliverState

Result: update local cache, if datafile of notifications has sent completely and local cache updated successfully then Meta Cache will be evicted.



3. Evictor:

Evictor is modified WS model component has been used to evict the leased used data from LCC based on modified LFU algorithm. LFU more suitable with web objects rather than other evicts algorithms as LRU (Ranchal et al., 2018, 212).

2.4 Evictor Functionality

The evictor applied the general idea of LFU (Leased Frequently Used) algorithm, which evict the least accessed objects (Mostefai et al., 2025). The

Evictor has been developed to replace the least frequently data with new one.

The proposed solution identifies the least-used records and is compatible with JSON files by introducing a new attribute for each record, which represents the last activity date. Consequently, the records with less active will be deleted. The strategy of replacement in modified WS model applies LFU, when the LCC size reaches to N size.

2.5 Evictor Contribution

Evictor contribute within modified WS model as:

1. Save LCC space
2. Reduce RT

3. GUI Responsiveness Enhancer (GUIRE)

GUI Responsiveness Enhancer: this component has been designed based on multitasking technique that contributes to reduce waiting time

This component depends on asynchronous tasks, which increases range of responsiveness. *Asynchrony* is essential for process that are potentially blocking such as the process that depend on web resources, so if the process is blocked with web resource then the other work can continue (Microsoft, 2018).

Waiting/Freezing time is consumed time, while processes wait to another process to be executed (Silberschatz et al., 2013). WS needs relatively long time to be executed, this time depend on many factors such as: internet speed, WS servers specifications, client specifications, network traffic, resources availability. These factors affect negatively or positively on the execution time of WS. When the consumer (user, system, application) requests WS from WS Server Provider, the user application waits in freeze mode until the response done, so during this time the user can't interact with the application.

3.1 GUIRE Functionality:

GUIRE makes ELMWS client side processes work in parallel based on multitasking.

Thus, all processes operate separately in asynchronous mode in the same System or Application. *Consequently*, all processes will be active, and never wait to the currently blocking task until finish.

3.2 GUIRE Contribution in ELMWS

There are many scenarios that make GUI go through wait/freeze mode such as:

- 1- Access to WS, this scenario happens when the data not found in local cache
- 2- Get the notifications from WS.
- 3- Access to WS with low speed of internet
- 4- Invoke WS without any response from WS Server due to server problems.

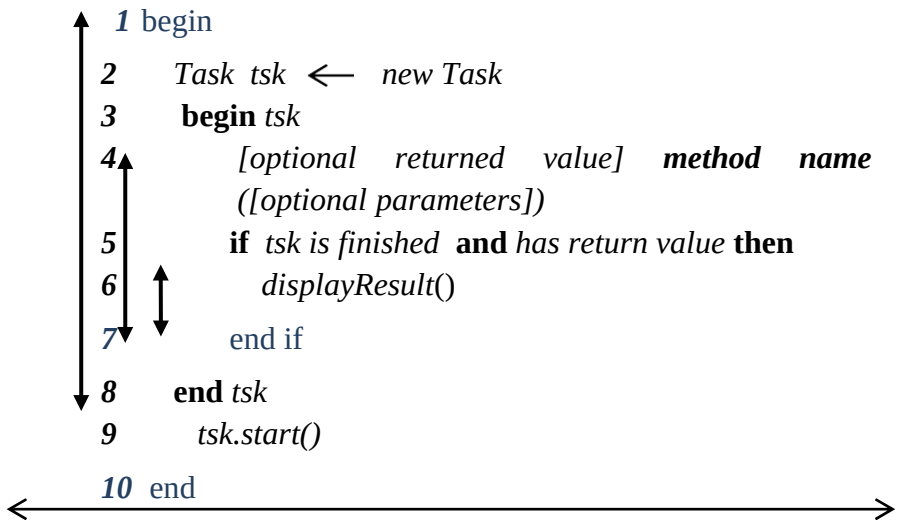
All these scenarios make GUI effectiveness and interaction very weak and slow, so the client application will go in freeze mode for some time (hung). Based on process execution time, but when GUIRE is applied, it contributes to eliminate waiting time, **consequently**, the responsiveness and interaction with ELMWS are increased, so this lead to reduce boring wait of users.

3.3 GUIRE Algorithm:

Algorithm: GUIRE

Input Data: method name

Result: asynchronous method with other methods



The algorithm above shows how GUIRE works, **first**, start with creates new Task.

Second, call a method that is needed to be in synchronous mode. **Third**,

check if task finished then it will be ready to display data when method has return value. **Finally**, start the task.

1. Experiments and Results:

At this section, the experiments have been done to measure and test ratio enhancement between search operation on Local client Cash (LCC) and search from WS Provider. Consequently, these experiments aim to demonstrate the added components to the model improves the performance of operations such as search, compared to performing the search directly from the service source, as is done in traditional web services models. The operation of search has done as Batch search. This mean the whole file content will be fetch and display in client monitor. In these experiments vital factors will be measured such as:

- 1- **Response Time (RT):** is defined as the total elapsed time between sending a service request and receiving the corresponding response, and it is considered a critical metric for evaluating web service performance (TechTarget, 2025).
- 2- **CPU Utilization:** is a key indicator of system efficiency, as excessive processor usage may negatively affect scalability and responsiveness of web services (Microsoft, 2023).
- 3- **Wait/Freez:** Asynchronous programming techniques, such as async and await, enable non-blocking execution of tasks, thereby improving application responsiveness and preventing user interface freezing (Microsoft, 2018).

3.1 Data Used:

Sample of data that has been used:

JSON Record= {"ctid":1,"ctname":"Advanced Research on Modern Systems","ctcat":"Information Technology","ctauthor":"Dr. JAMAL Alawee","ctpath":"/storage/data/topics/it/archive_001.pdf","ctdate":"2026-01-16"}

Table (1):Dataset description of JSON files

No. of File	Size /MB	Approximately Count of Records
1	1	4,810
2	10	48,100
3	20	96,200
4	30	144,300
5	40	192,400
6	50	240,500

1. Measure Response Time (RT):

In next experiments the RT factor has been measured for LCC and WS.

Experiment 1: Measure LCC vs. WS RT of batch search

This experiment has been conducted for 6 times, each time is performed on dataset as in table (1). In this experiment, evaluation of the performance depending on RT is done. A comparison shows the difference between RT of search from LCC and WS provider database.

Input: 6 search operations as Batch search on JSON file with 1000 record.

Result: comparison between RT of LCC and Web service Batch search

Time Unit: Seconds

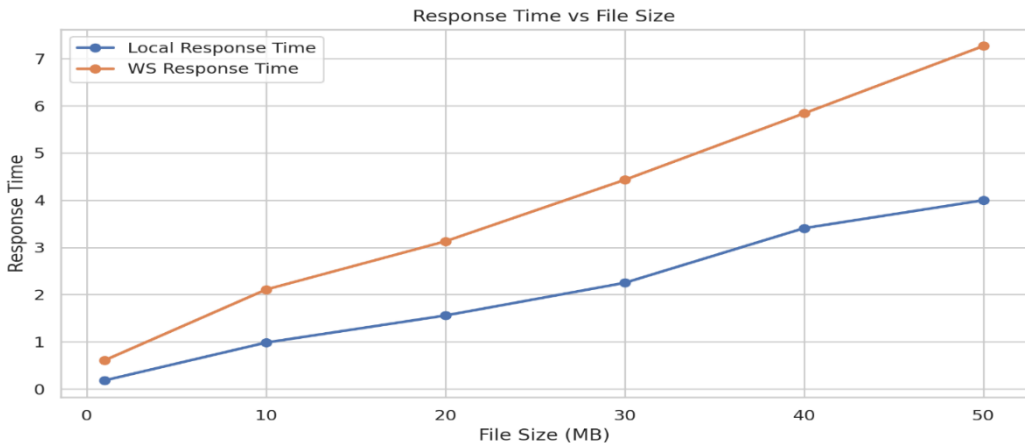


Figure (6): Measuring RT for Batch search from LC and WS

Result Analysis:

The previous Figure clarifies the performance of RT within ELMWS. There is a clear time gap between search from LC and search from WS within a

server using WS.

It has been noticed that the LCC enhanced the ELMWS performance significantly.

Overall, RT showed about 47.1% improvement over WS RT.

2. *Measure CPU Utilization Factor*

The experiments are done to measure extend impacting of ELMWS on CPU Utilization factor.

Experiment 1: *LCC vs.WS CPU Utilization of batch search*

In this experiment, CPU Utilization has been conducted for 6 times, each time is performed on dataset as in table (1) . This test has been done on search from LCC and WS. The results appear in Figure (7).

Input: 6 search operations as Batch search, OJSON files

Result: comparison between CPU Utilization of Local cache and Web service Batch search

Time Unit: Seconds.

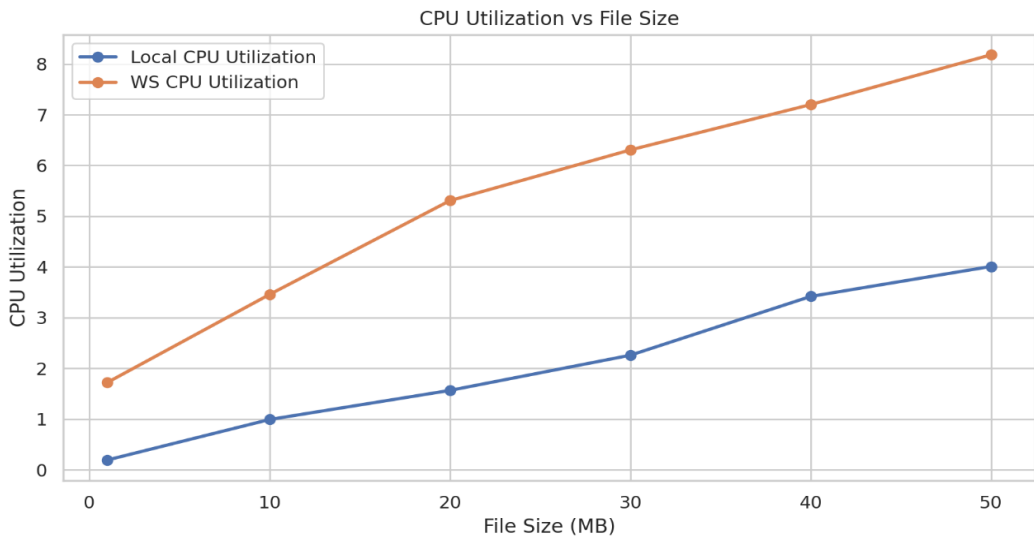


Figure (7): Results of CPU Utilization of LCC and WS batch search

Result Analysis:

Based on previous Figure the CPU Utilization of searching operations has been reduced, when they are done on LCC compared to WS CPU Utilization.

Overall, **CPU Utilization showed about 61.3% improvement over WS CPU Utilization.**

3. Measure Waiting/Freezing Time

This experiment measures Waiting/Freezing before and after apply GUIRE on ELMWS. The experiment has been conducted 10 times. This experiment applied on Optimizing JSON files and search from WS providers at the same time.

Experiment 1: Measure Waiting Time with GUIRE.

Input: invoke WS to create Local cache, delivered JSON File as LCC.

Result: create JSON Local cache with size 50MB in client side with 10 separated times, measuring Wait Time

Time Unit: Seconds.

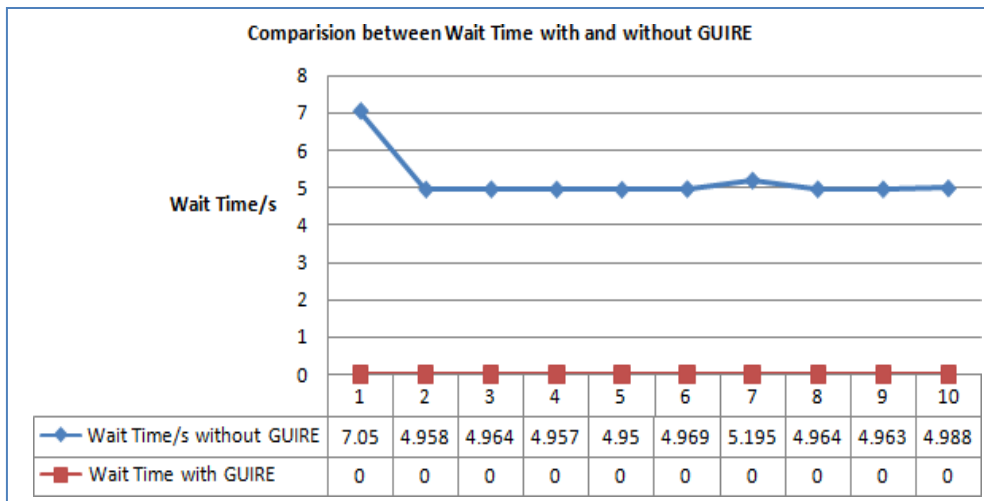


Figure (8): Measure waiting time and Asynchronous time of WS process

Result Analysis:

The previous Figure demonstrate the enhancement of ELMWS performance based on GUIRE. The waiting time of both GUI and Application processes equal zero, while WS has been executed. *Consequently*, ELMWS increases the interaction between users and ELMWS as well as reduces the

feeling of boredom while waiting to execute operations.

Conclusion:

The proposed ELWSM has been developed and tested successfully. The ELWSM has many efficient components include: *CCM*, *LCC*, *GUIRE* and *Evictor*. Based on results analysis, the ELWSM saves the significant system resources by reducing RT by 47.1%, CPU Utilization by 61.3% and Waiting time by 99%. Consequently, applying the proposed ELMWS on E-learning systems will ensure stability, resilience, availability, and reliability of ELMWS ,refer to acquired results. The bellow table(2) meet matching components and findings.

Table (2):matching Components and ELMWS Objectives

	LCC	DC	GUIRE
Stability	Reduce RT, Reduce CPU Utilization		Reduce Response Time, Waiting Time
Availability	Reduce Memory space		Reduce Response Time, Waiting Time
Reliability		Ensure delivery	

Overall, the findings confirm the effectiveness of the proposed model in achieving high-performance and robust web service management in e-learning environments.

Future Trends:

Refer to previous ELMWS, enhancement of security issues is essential trend. Thus, the performance of ELMWS will be improved and trusted.

References:

- Ali, W., Shamsuddin, S. M., & Ismail, A. S. (2021). A survey of web caching and prefetching techniques. *International Journal of Computer Science and Network Security*, 21(3), 45–56.
- Papazoglou, M. P., & Van Den Heuvel, W. J. (2020). Service-oriented design and development methodology. *International Journal of Web Engineering and Technology*, 15(1), 1–28.
- Pautasso, C., Zimmermann, O., & Leymann, F. (2021). RESTful web services vs. SOAP: Principles and trade-offs. *IEEE Software*, 38(2), 72–79.

- Dustdar, S., Leymann, F., Papazoglou, M. P., & Schreiner, W. (2022). Principles of service-oriented computing. *IEEE Internet Computing*, 26(1), 8–17.
- AltexSoft. (2025). *Web services: Use cases and key architectures*. <https://www.altexsoft.com/blog/web-service/>.
- TechTarget. (2025). *What is response time?* <https://www.techtarget.com/...https://www.techtarget.com/searchnetworking/definition/response-time>
- Zhang, Y., Liu, H., & Wang, L. (2023). Enhancing Web Service Performance through Caching Strategies. *IEEE Transactions on Services Computing*.
- Microsoft. (2023). Interpreting CPU utilization for performance analysis. <https://learn.microsoft.com/...>
- Halim, M., Dhelim, S., Hammia, A., & Balla, A. (2020). Web services composition in e-learning platforms. *International Journal of Emerging Trends in Engineering Research*, 8(2), 418–425.
- Mostefai, S., Bourouis, A., & Ahmed-Nacer, M. (2025). Cache Replacement Algorithms for Web Services: A Comparative Study. *Journal of Network and Computer Applications*.
- Mostefai, B., Boutefara, T., Bousbia, N., Dhelim, S., & Hammia, A. (2025). Enhancing user experience in e-learning systems: A user-centric RESTful web services approach. *Computers in Human Behavior Reports*, 10,
- Microsoft. (2018). Asynchronous programming with async and await. <https://learn.microsoft.com/...>
- Julien Gascon-Samson, M.C., Fan Jin, Jörg Kienzle, Bettina Kemme, *CacheDOCS: A Dynamic Key- Value Object Caching Service*. IEEE, 2017.
- Ali, W., Shamsuddin, S. M., & Ismail, A. S. (2020). Caching strategy for Web application – a systematic literature review. *International Journal of Web Information Systems*, 16(5), 545–573. doi.org
- Goethem, L. V., Sanchez-Rola, I., Meng, W., Huygens, C., & Joosen, W. (2020). Abusing browser cache headers to identify and track online users. *Proceedings of the 21st Privacy Enhancing Technologies Symposium (PETS 2021)*, 33–52. petsymposium.org

- Kommareddy, R. R. (2025). Design patterns for multi-tiered caching in high-throughput web applications. *International Journal of Creative Research Thoughts*, 13(5), 1236–1242. www.ijcrt.org
- Mansuri, F. (2025). *Caching strategies for web apps: From browser to backend*.
- Osmani, A. (2025). *Double-keyed caching: How browser cache partitioning changed the web*. AddyOsmani.com.
- Wibowo, A., et al. (2025). Implementation of cache memory technology in improving performance of modern computing systems. *Jurnal Penelitian Pendidikan IPA*, 11(6).
- Rohit Ranchal, B.B., Pelin Angin, Lotfi ben Othmane, *EPICS-A Framework for Enforcing Security Policies in Composite Web Services*. IEEE, 2018.
- Mostefai, B., Boutefara, T., Bousbia, N., Balla, A., Dhelim, S., & Hammia, A. (2025). Enhancing user experience in e-learning systems: A user-centric RESTful web services approach. *Computers in Human Behavior Reports*.
- Waller, M. A., Fawcett, S. E., & others. (2013). Data science, predictive analytics, and big data: A revolution that will transform supply chain design and management. *Journal of Business Logistics*, 34(2), 77–84.
- Silberschatz, A., Galvin, P. B., & Gagne, G. (2013). *Operating system concepts* (9th ed.). Wiley