Journal of King Saud University - Computer and Information Sciences 36 (2024) 102201

Contents lists available at ScienceDirect

Journal of King Saud University - Computer and Information Sciences

journal homepage: www.sciencedirect.com

Software requirement engineering over the federated environment in distributed software development process

Abdulaziz Alhumam, Shakeel Ahmed *

Department of Computer Science, College of Computer Sciences and Information Technology, King Faisal University, Al-Ahsa 31982, Saudi Arabia

ARTICLE INFO

Keywords:
Software Engineering
Federated Learning
Software Requirement Engineering
Feature Engineering

ABSTRACT

In the recent past, the distributed software development (DSD) process has become increasingly prevalent with the rapid evolution of the software development process. This transformation would necessitate a robust framework for software requirement engineering (SRE) to work in federated environments. Using the federated environment, multiple independent software entities would work together to develop software, often across organizations and geographical borders. The decentralized structure of the federated architecture makes requirement elicitation, analysis, specification, validation, and administration more effective. The proposed model emphasizes flexibility and agility, leveraging the collaboration of multiple localized models within a diversified development framework. This collaborative approach is designed to integrate the strengths of each local process, ultimately resulting in the creation of a robust software prototype. The performance of the proposed DSD model is evaluated using two case studies on the E-Commerce website and the Learning Management system. The proposed model is analyzed by considering divergent functional and non-functional requirements for each of the case studies and analyzing the performance using standardized metrics like mean square error (MSE), mean absolute error (MAE), and Pearson Correlation Coefficient (PCC). It is observed that the proposed model exhibited a reasonable performance with an MSE value of 0.12 and 0.153 for both functional and non-functional requirements, respectively, and an MAE value of 0.222 and 0.232 for both functional and non-functional requirements, respectively.

1. Introduction

In recent times, software engineering has undergone a profound transformation concerning requirement engineering, fault localization, benchmarking, code translation, dependency evaluation, troubleshooting, and adoption of distributed software development. Distributed software development (Jabangwe et al., 2016) refers to the collaborative development of software systems by geographically dispersed teams, often operating in different time zones and organizational structures. With the advent of the Federated learning environment, software development has taken remarkable innovative possibilities like scalability, distributed model training, privacy preservation, and rapid development process (Xu et al., 2023). This approach addresses the challenges arising from having development teams spread out across distinct projects of the prototype, sometimes using diverse development processes and having various stakeholder perspectives. SRE utilizes federated environments to create a cohesive ecosystem that surpasses geographical borders, enabling the unified global model. The global model elucidates the requirement prioritization for better prototypes.

The DSD allows multiple entities to work collaboratively on the same project across different interdependent components simultaneously (L'Erario et al., 2020). Unifying multiple interdependent components would be a challenging task, as the priority of the requirements does change from component to component. The priorities across the components of the software prototype must be considered while building the global model. Those priorities have to be normalized while assigning a priority to the feature, i.e., the requirement in the global model (Qi et al., 2024). The simultaneous development of the components would assist in faster prototype development. The federated learning environment on the top ensures the security of the sensitive data associated with the software requirement, design, and prototyping (El Koshiry et al., 2024). Furthermore, distributed development mitigates the risks associated with single-point failures, as redundancy and fault tolerance are inherently built into the process.

* Corresponding author at: Department of Computer Science, College of Computer Sciences and Information Technology, King Faisal University, PO BOX: 400, Hofuf 31982, Kingdom of Saudi Arabia.
E-mail addresses: aahumam@kfu.edu.sa (A. Alhumam), shakeel@kfu.edu.sa (S. Ahmed).

https://doi.org/10.1016/j.jksuci.2024.102201
Received 26 June 2024; Received in revised form 17 September 2024; Accepted 25 September 2024
Available online 28 September 2024

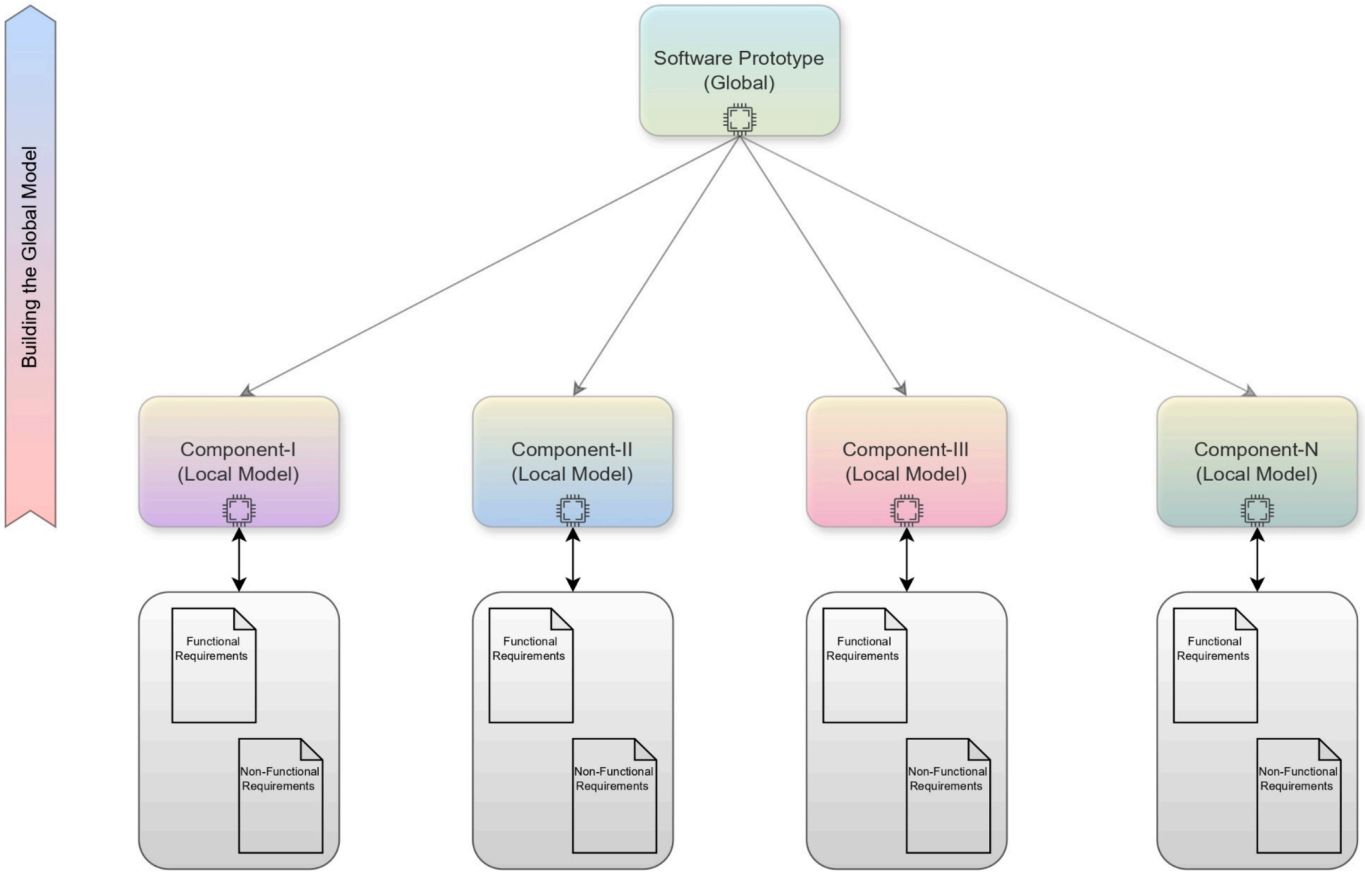

Fig. 1. Distributed Software Development in a Federated Learning Environment.

DSD has risen in popularity in recent years as a result of globalization, remote work patterns, and the requirement for specialized expertise from various geographical areas. A federated environment, defined as remotely connected, independent modules that work together, is often used to assist such DSD processes and yield a better software prototype. The primary challenge in the DSD is developing a global software prototype from multiple local prototypes by considering divergent software requirements. Each software model would have a divergent requirement, and they are weighted accordingly based on their significance and the engineering technique required. Integrating all such requirements and software development models needs a robust model to effectively address the constraints associated with requirement prioritization (Mostafa et al., 2024). The current study focuses on developing a global model that can effectively develop a global model by integrating all the local models and normalizing the requirement's significance across the models.

Federated learning greatly improves security in the software development process through the decentralization of data storage and processing. Instead of aggregating data on a central server, federated learning allows individual devices to train models locally using their datasets (Wen et al., 2023). This method reduces the likelihood of data breaches and unauthorized access by keeping sensitive information on local devices and avoiding transmission of actual data over the internet. Rather, they exchange metadata like feature significance-related data. Techniques like differential privacy ensure that even the updates shared with the central server do not reveal software prototype-related data. This approach makes software development more secure, protects user privacy, and meets data protection regulations. The federated learning technology is a cost-effective way of developing the project in an agile development process, minimizing the need for extensive centralized infrastructure and reducing data transfer costs (Khan et al., 2023). The idea of the current DSD model in federated learning is shown in Fig. 1. It can be observed from the figure that multiple local models used in developing the components of the prototype are integrated over a federated learning environment to develop the global model. The research questions (RQ) analyzed across the study are listed below.

RQ1: Can the simultaneous development of the components of the software prototype be feasible concerning ever-changing software requirements (SR).

RQ2: The significance of the requirements would differ from component to component of software, and a unified requirement weight has to be assessed for optimized prioritization of the requirements in the global model.

RQ3: The development time and cost of a global model in the federated learning environment are comparatively better than the conventional DSD models.

There are certain assumptions made in the current study for ease of implementation and to confine the study to specific aspects of software engineering. The assumptions that are made in the current study are listed below.

Assumption 1. The requirements in the software are considered as the features, where each requirement is considered as a single feature. These features are processed to build the software prototype optimally.

Assumption 2:. The feature weights are assessed using different feature engineering techniques like Stochastic Gradient Descent (SGD) and Ant Colony Optimization (ACO), which are used in the local models.

Assumption 3. To evaluate the working of the federated learning model, the feature weights are aggregated to the global model based on the feature weights being evaluated at the local models.

The current study is motivated by the necessity of building a software

prototype in the distributed framework for the simultaneous building of the components of the prototype at different geographical locations. Federated learning is considered to be an efficient and secure way of building software in a distributed environment (Cui et al., 2022). Furthermore, distributed development improves the ability to handle larger workloads and increases the ability to adapt rapidly to evolving requirements for the project. It also allows for incorporating changes to prototypes from different local models. Including federated learning guarantees data privacy and security since confidential information is confined to local sources, minimizing the possibility of breaches, thereby contributing to a robust software prototype (Le et al., 2023; Alshehri et al., 2024). The contributions of the current study are listed below.

- The functional and non-functional requirements associated with the desired software prototype considered in the case study are being identified.
- The requirements are considered as the features, and the weights to those features are assigned using the feature weight assessment technique.
- The assigned weights are updated over the iteration, and the optimized weights are aggregated to the global model.
- The performance of the proposed federated learning-based software development model is assessed concerning various evaluation metrics.

The rest of the sections of the manuscript are arranged in the following manner. Section 2 discusses the related work in distributed software development. The section also outlines the technical gaps associated with the conventional DSD models. Section three presents the background of the current study, where the feature engineering, dataset details, and implementation environment are discussed. Section four presents the proposed federated learning-based distributed software prototyping. Section five presents the experimental analysis of the proposed model. Finally, section six presents the conclusion and the future research directions.

2. Literature review

There are considerable studies on distributed software development technology, as the DSD would yield a more efficient, robust, and scalable prototype in a much more cost-effective and secure manner. This variety of locations also makes it possible for continuous development processes since teams in different time zones can work on the project continuously, which can also significantly reduce the development time (Taweel and Brereton, 2006). These are some of the reasons for the wider acceptance of DSD models. The existing studies in DSD and federated learning are discussed in the current section of the manuscript.

Marum Simão Filho et al. (Filho et al., 2018) have done a systematic review on task allocation in DSD, and have stated that distributed development is crucial in the software industry. The study has outlined multi-criteria models for task assignment in distributed software development, highlighting qualitative decision-making methods and identifying key aspects and classifications that could benefit future research. A study based on Agile Software Development (David et al., 2023) in a distributed environment has stated Agile teams emphasize face-to-face communication and collaboration, but distributed software development challenges this due to spatial, temporal, and cultural separations. A Grounded Theory study involving 55 participants from 38 software companies in the USA, India, and Australia reveals that distributed Agile teams require substantial senior management support in areas such as organizational culture, HR management, financial sponsorship, infrastructure, technology, and customer liaison. Another study by Junior David et al. (David et al., 2023) on the topic of the Agile Management Model for Distributed Software Development (AgiTeD), has stated that it enhances distributed team support, particularly through the Agile Leader role and the Repository and Communication Rules artifact, on conducting the survey for 23 software developers who work remotely.

M. S. Farooq et al. (Farooq et al., 2022) in the study on the blockchain-based framework for Agile-based software prototyping aimed at addressing transparency, trust, security, and traceability issues in DSD. By utilizing smart contracts on a private Ethereum blockchain for tasks such as acceptance testing, secure payment, and penalty assignment, AgilePlus aims to enhance communication coordination, and mitigate trust issues between customers and development teams while also addressing scalability challenges through the use of Interplanetary File System (IPFS) for off-chain storage. The Agile Development Cloud Computing framework (ADCC), as studied by Younas et al. (Younas et al., 2020), integrates agile development with cloud computing to support local and remote agile development environments. While assessment findings have shown that cloud computing within the ADCC framework effectively addresses face-to-face communication, transparency, scalability, and resource management issues, it still presents hurdles regarding data security threats, privacy concerns, interoperability, and high costs. Furthermore, ADCC does not own blockchain technology to address traceability, security, and trust concerns.

There are few studies on using federated learning in the software engineering domain. A case study by Yanming et al. (Yang et al., 2024) using Federated Learning for Software Engineering for Code Clone Detection and Defect Prediction. The study has summarized that a federated learning-based framework addresses these challenges by enhancing model performance on skewed industrial data while ensuring privacy, demonstrating superiority over baseline methods in real-world scenarios. Alharbi et al. (Alharbi et al., 2022) have presented an empirical investigation of distributed software testing using a Fuzzy TOPSIS-based approach. The study has analyzed the challenges IT businesses face in remote software production due to the COVID-19 pandemic, emphasizing the importance of software testing for performance and accuracy. Using the Fuzzy TOPSIS method, it prioritized ten identified software testing challenges, acknowledging limitations such as potential bias and the fixed number of issues considered and suggesting future research to explore other multi-criteria decision-making approaches and weight estimation techniques.

A survey on interaction design in DSD models was presented by Domingos Alves et al. (Domingos Alves et al., 2023) to examine how distributed teams handle interaction design tasks and the complexities that arise from geographical, temporal, and cultural differences. In addition to identifying critical behaviors that may enhance teamwork and project results, the research also highlights major obstacles these teams encounter in the DSD model. Another study on learning soft skills through DSD (Alharbi et al., 2022) involves a distributed online course where Belarusian university students worked on industrial projects for Danish clients. The course aimed to teach Scrum and enhance soft skills such as communication, teamwork, and problem-solving. The course also provided Belarusian teachers with insights for future course development.

Conventional distributed software development encounters certain constraints that impede its efficiency and effectiveness. A significant constraint is the presence of communication obstacles resulting from geographical and time zone disparities, which may result in misinterpretations and project timetable delays. In addition, ensuring similar quality and standards across geographically separated teams is a challenge due to differences in experience and local practices, which might impact the uniformity of the end result. Security issues are also a factor, especially when it comes to transmitting and storing data in different places, which raises the likelihood of data breaches. Federated learning can address many of the limitations in conventional DSD by enhancing collaboration, security, and efficiency. By allowing multiple decentralized devices or servers to collaboratively train machine learning models without sharing raw data, federated learning

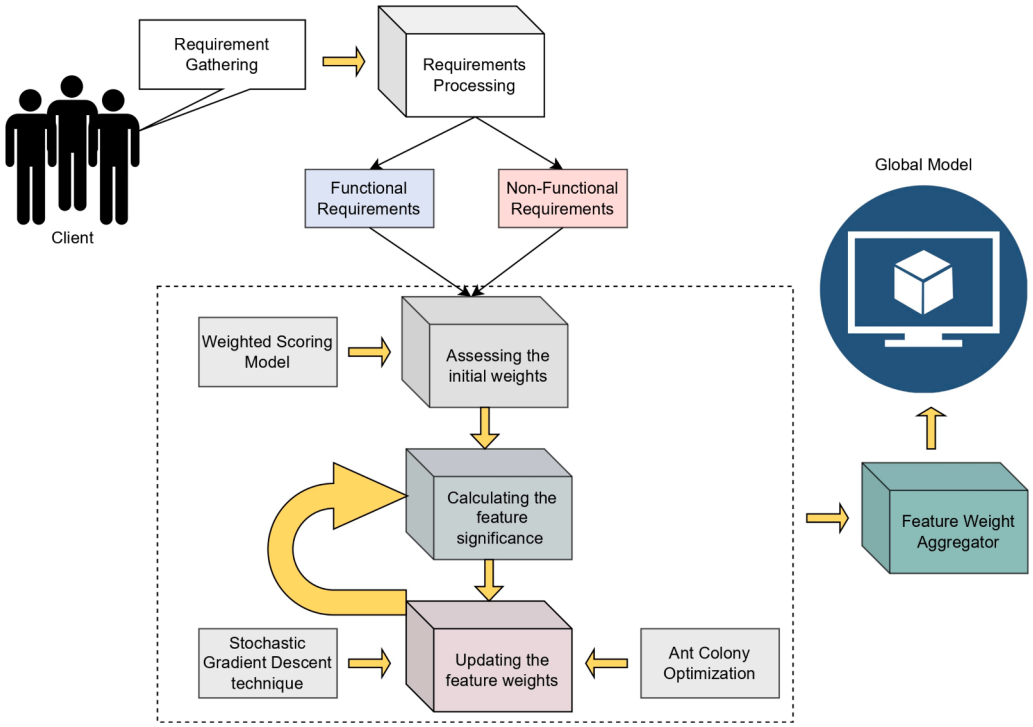

Fig. 2. The workflow of the proposed DSD model in the FL environment.

Software Requirements					
Functional Requirements		Non-Functional Requirements		Non-Functional Requirements	
User Account Creation	User Authentication	Higher Significance		Lower Significance	
User Profile Management	Product Catalog				
Product Reviews	Shopping Basket	Data Privacy	Compliance	Load Handling	Disaster Recovery
Order Management	Payment Processing	Scalability	Availability	Localization	Interoperability
Customer Support	Inventory Management	Activity Log	Reliability	Customization	Extensibility
Course Management	Teacher-Course Mapping	Maintainability	Responsiveness	Audibility	Modularity
Student-Course Mapping	Course Organizer	Cross-Browser Compatibility	Latency	Version Control	Documentation
Progress Tracker	Assessment Module	Resource Management	Third-Party Integration	Data Backup	Personalization

Fig. 3. The functional and non-functional requirements are considered in the current study.

significantly reduces the risk of data breaches and enhances privacy.

3. Material and methods

The current section of the manuscript discusses the data that is being considered in evaluating the model, details of the implementation environment, and pre-processing techniques are being discussed. Feature engineering plays a vital role in the current study. The software requirements are considered features and processed in the federated environment, which is demonstrated using the two case studies discussed in a subsection of the current study. The working process in the current study is shown using the flow diagram, as shown in Fig. 2. The requirements gathered from the clients are processed and segregated into functional requirements (FR) and non-functional requirements (NFR). NFRs are not explicitly defined, but they are identified based on the FR and the nature of the prototype that is being developed.

Table 1
Details of the Implementation Environment.

Environment	Details
Machine	Standalone Laptop
Make	Dell Inspiron 3530
Processor	Intel Core i7 13th Gen
RAM	16 GB DDR3
Operating System	Windows 11
Coding Platform	Python
Implementation Platform	Jupiter Notebook v7.1.2
Libraries	TensorFlow, PyTorch, NumPy, Flask

Upon processing the requirements, the initial weights of the features are assessed, and those feature weights are updated using the techniques that are being used in the local models. Finally, those feature weights are aggregated by the feature aggregator, and those aggregated feature weights are used to develop the global model.

3.1. Data collection and processing

The data in the current study is determined based on the FR and NFR associated with the case studies being considered. The initial feature weights of features are being considered based on their significance in the project. Functional requirements are considered highly significant, and non-functional requirements are prioritized at the next level (Shah, 2016; Shankar et al., 2020). Subsequently, the feature weights are optimized using the feature engineering mechanism. The features that are considered in the current study are categorized as shown in Fig. 3. The lower significant features do not mean that they can be ignored during the development process, rather, they are given lesser initial weights, and based on their impact on the development process, they are given more weightage over the iterations. All the components that are desired to be the functionalities of the prototype are considered as the function requirements, where the features can be made in the prototype (Siddique et al., 2024).

On the other hand, the non-functional requirements are those that will not be directly accessible to the user, but they would assist in better robust software, and they are given considerable weightage as functional requirements are more or less dependent on the non-functional requirements (Raj Kumar Chopra, 2016). Not all non-functional requirements are considered in the software development process; they are prioritized based on their significance to the prototype being built. Furthermore, the features are assigned feature weights and processed using feature-processing techniques (Ijaz et al., 2019).

The complete dataset is used as a single fold through the process weight estimation, updating, and aggregation. In bot processh the test cases that are being considered in the current study, 8 requirements for each FR and NFR are being considered in evaluating the proposed model.

3.2. Implementation environment

The current subsection of the manuscript presents the details of the implementation environment that is used in the current study. The model is evaluated using the standalone system that makes use of the pre-built libraries. As part of the evaluation, the local models are being implemented across multiple local machines, where two machines are considered the local model and one machine is considered the global model. The docker is used to integrate all the machines in the federated learning environment, and the models are locally implemented using the Jupiter notebook platform. The details of the machines are listed in Table 1.

3.3. Case studies

The proposed model is being evaluated using the case studies. Two case studies are being considered in the current study to evaluate engineering requirements in the federated learning environment. The E-commerce website project and the web-driven Learning Management System (LMS) are being considered in the current study. The desired functionalities are being discussed in the current study. After COVID-19, e-commerce and LMS have made a breakthrough in the market, as the majority of citizens across the globe wish to access the services remotely rather than move out of their homes (Costa and Rodrigues, 2023; Barona and Ramirez, 2021).

Fig. 4. Image representing the software requirements associated with the E-commerce website.

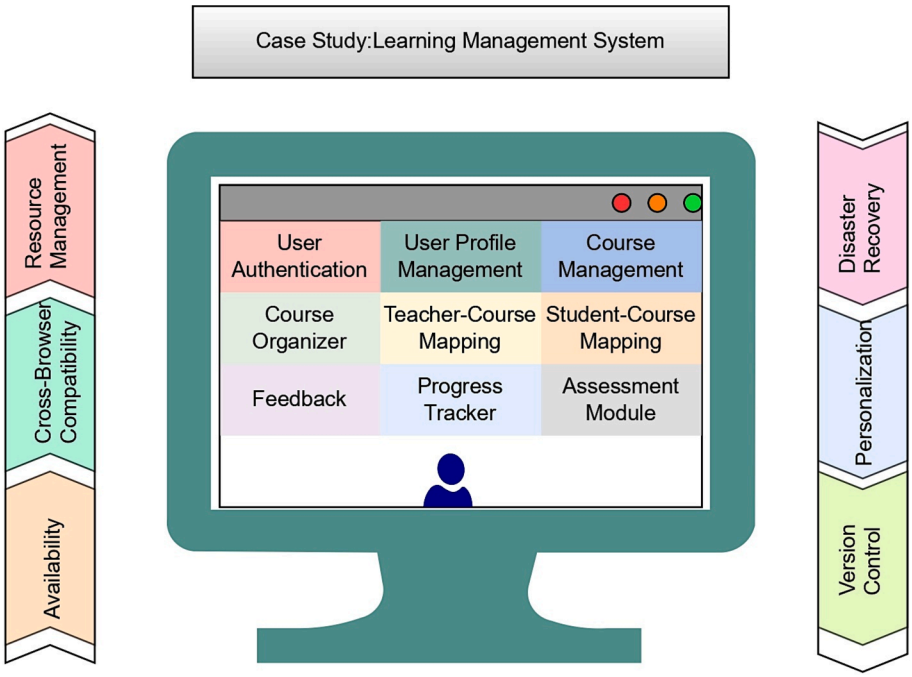

Fig. 5. Image representing the software requirements associated with the E-commerce website.

3.3.1. Case study on E-commerce website

E-commerce platforms have fundamentally transformed the retail industry, revolutionizing corporate operations and consumer shopping habits. The E-Commerce website functionalities are in common with various online platforms like amazon.com (Qin and Liu, 2022) and flipkart.com (Monsalve-Obreque et al., 2023), where the customers are allowed to browse across the variety of items that are sold through the website, and they can add them to the shopping basket and buy them from the website. The users are recommended to log in using valid credentials to the website before they place an order, and they are given the feasibility to update their account information, including the email ID, mobile number, address, and delivery preferences. In the payment process, users can pay online using cards or wallets and avail themselves of the cash-on-delivery option. All these are considered in the development of the e-commerce website, along with the aforementioned FR, and some of the NFRs are also being considered for the development of a robust prototype (Monfared and Kamandi, 2016). The corresponding requirements and functionalities for the E-Commerce website are shown in Fig. 4.

Apart from the abovementioned FR, which is desired to be the features associated with the e-commerce website, the NFR has to be considered while building a robust software prototype. The NFR includes the various additional functionalities that make sure the software prototype performs well and all the FRs appropriately function as desired. Moreover, the FR is dependent on the NFR for robust software. The FR and NFR are not limited to the ones shown in the figure above; they are various aspects that are being addressed in the NFR. The performance related aspects like data privacy, security, user-friendliness, scalability, maintainability, customization, interoperability, and extensibility are some of them that are not directly attributed by the customers, but NFR is essentially important like FR for proper functionality.

3.3.2. Case study on learning Management System

A Learning Management System is an application that is designed to automate the process of creating, delivering, and managing academic courses at universities and training institutions. It offers a centralized platform for faculty to post and organize course materials, monitor student progress, and evaluate performance. LMS systems consist of a variety of activities, including discussion forums, quizzes, assignments, and multimedia material, which facilitate a dynamic and captivating learning experience. Through LMS, educational organizations and universities can optimise administrative processes, promote interaction, and enhance the accessibility of instructional materials. The LMS would have some of the FR like user authentication, user management, course management, course mapping, Teacher-course mapping, student-course mapping, Progress tracker, Assessment module, feedback are some of the FR in the LMS project. The corresponding requirements and functionalities for the LMS are shown in Fig. 5.

Similar to the E-Commerce website, the LMS project will have more or less the same NFR, which is to be considered while building the prototype. Some of the NFRs include availability, cross-browser compatibility, resource management, version control, personalization, disaster recovery, backup management, documentation are few of them. The NFR would be common for almost all the projects, as they are concerned to the performance related requirements of the software prototype. When the NFR is less relevant to the project, the feature weight is assumed to be almost zero.

3.4. Evaluation metrics for the model

The performance of the FL-based DSD model is largely dependent on the feature weights that are being assigned. The more weight is assigned to the feature, the more significant the requirement is. The performance of the proposed model is evaluated using loss measures like the Mean Squared Error, Mean Absolute Error (Emmanuel et al., 2021), and the Pearson Correlation Coefficient (Li et al., May 2022) are being used in the evaluation of the proposed model.

Mean Squared Error approximates the average squared error between the actual value and the predicted value. For over n observations, the actual value av and predicted value pv are used in assessing the error as shown in Equation (1).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(av_i - pv_i)^2 \tag{1}$$

Mean Absolute Error is almost identical to the MSE, which assesses the magnitude of error between the actual value and the predicted value.

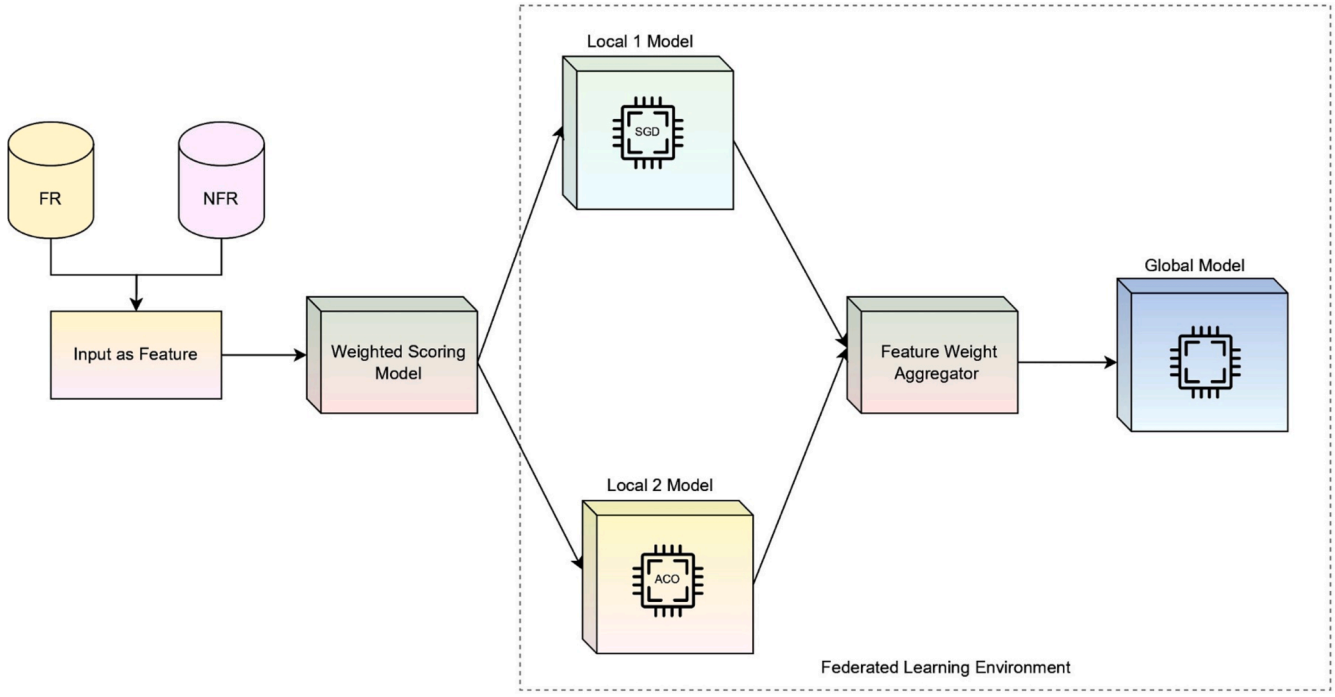

Fig. 6. Block diagram of the proposed FL-based DSD model.

For over n observations, the actual value av and predicted value pv are used in assessing the error as shown in Equation (2). The MAE performs comparatively better in the case of outliers as it takes the absolute value of errors.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|av_i - pv_i| \tag{2}$$

Pearson Correlation Coefficient quantifies the linear relation between the two continuous variables, it depicts the strength and direction of the relation between two variables. When the resultant outcome is positive, it indicates positive linear relation among the variables, when the corresponding resultant value is zero, both the variables are independent of each other. When the resultant value is negative, i.e., less than zero, it indicates a negative linear relation among the variables. The corresponding formula for the PCC over two independent variables x, y is shown in Equation (3).

$$PCC = \frac{\sum_{i=1}^{n}(x_i - x')(y_i - y')}{\sqrt{\sum_{i=1}^{n}(x_i - x')^2 \sum_{i=1}^{n}(y_i - y')^2}} \tag{3}$$

From the above equation, the x_i represents the i^{th} instance of the x, and notation x' is the mean of all the samples of x. The abovementioned metrics are used to evaluate the proposed model in terms of feature significance. Similarly, the y_i represents the i^{th} instance of the y, and notation y' is the mean of all the samples of y. The above discussed are some of the metrics that are used in evaluating the proposed model.

4. Feature engineering

The requirements that are associated with the prototype are assumed to be the features, and the feature engineering is significant in determining which specific requirement is of higher priority and has to be incorporated before implementing the rest (Mamdouh Farghaly and Abd El-Hafeez, 2023). Initially, the feature weights are assigned using the Weighted Scoring Model (WSM) approach, and over the iterations, the feature weights are updated. The FR is generally given more weight than the NFR, based on customer preferences. However, not all the FR would have a higher weightage than the NFR. The block diagram in Fig. 6 represents the overall working of the proposed model.

Table 2
The influential factor for each of the aspects of project development.

Aspect	Influential Factor
Cost to Implement	0.2
Impact on Client	0.3
Risk Factor on Implementation	0.1
Project Dependency	0.2
Business Impact	0.2

Table 3
The initial weights associated with Functional Requirements.

Requirement	CoI	IoC	RFI	PD	BI	Final Weight
User Account Creation	2 × 0.2	3 × 0.3	2 × 0.1	1 × 0.2	3 × 0.2	2.3
User Authentication	2 × 0.2	3 × 0.3	2 × 0.1	1 × 0.2	2 × 0.2	2.1
Product Catalog	3 × 0.2	3 × 0.3	3 × 0.1	2 × 0.2	2 × 0.2	2.6
Product Reviews	2 × 0.2	2 × 0.3	2 × 0.1	1 × 0.2	2 × 0.2	1.8
Payment Processing	3 × 0.2	3 × 0.3	3 × 0.1	3 × 0.2	3 × 0.2	3
Customer Support	2 × 0.2	2 × 0.3	2 × 0.1	2 × 0.2	3 × 0.2	2.2
Course Management	3 × 0.2	3 × 0.3	3 × 0.1	2 × 0.2	3 × 0.2	2.8
Course Tracker	2 × 0.2	3 × 0.3	1 × 0.1	1 × 0.2	2 × 0.2	2.0

4.1. Initial feature weight assessment using WSM

The initial feature weights are exceptionally significant in robust weight assessment at the subsequent rounds and faster convergence of the model (Mamdouh Farghaly and Abd, 2022; Siddiqa, 2024). Rather

Table 4
The initial weights associated with Functional Requirements.

Requirement	CoI	IoC	RFI	PD	BI	Final Weight
Data Privacy	3 × 0.2	3 × 0.3	3 × 0.1	2 × 0.2	3 × 0.2	0.84
Scalability	3 × 0.2	2 × 0.3	3 × 0.1	1 × 0.2	2 × 0.2	0.63
Interoperability	3 × 0.2	3 × 0.3	3 × 0.1	2 × 0.2	2 × 0.2	0.78
Compliance	3 × 0.2	1 × 0.3	2 × 0.1	0 × 0.2	1 × 0.2	0.39
Maintainability	3 × 0.2	0 × 0.3	3 × 0.1	1 × 0.2	2 × 0.2	0.45
Extensibility	2 × 0.2	0 × 0.3	2 × 0.1	1 × 0.2	3 × 0.2	0.42
Documentation	1 × 0.2	0 × 0.3	1 × 0.1	0 × 0.2	1 × 0.2	0.15
Personalization	3 × 0.2	3 × 0.3	3 × 0.1	1 × 0.2	3 × 0.2	0.78

than randomly assigning the weights, which would result in an inappropriate approximation of the weights (Guleria et al., 2022). The use of WSM in initial weight assessment would result in better-optimized prioritization, objective criteria, Cross-functional alignment, and transparency of the weight assignment process (Jadhav and Sonar, 2009). While the weights are assigned to the features, various aspects like Cost to Implement (CoI), Impact on Client (IoC), Risk Factor on Implementation(RFI), Project Dependency (PD), and Business Impact (BI). Each of them is given a significant weight based on the dependability of the project. The influential factor of each aspect is shown in Table 2. The assigned weights for each of the FR and NFR requirements are shown in the table below using the WSM approach shown in Tables 3 and 4, respectively.

The formula for assessing the initial weights I_ω The functional and non-functional requirements are presented in Equations (4) and (5), respectively. The coefficients for each of the aspects are as follows the notation a associated with CoI, the notation b associated with IoC, the notation c associated with RFI, the notation d associated with PD, and e is associated with BI.

$$I_\omega(FR) = a \times 0.2 + b \times 0.3 + c \times 0.1 + d \times 0.2 + e \times 0.2 \quad (4)$$

$$I_\omega(NFR) = \lambda \times \{a \times 0.2 + b \times 0.3 + c \times 0.1 + d \times 0.2 + e \times 0.2 \quad (5)$$

From the above equation, the notation λ is the bias term that is considered 0.3 in the current study. The NFR has a comparatively lesser initial weight than the weights assigned to FR.

The above are the initial weights that are associated with some of the project requirements.

4.2. Feature weight updating using SGD

The initial weights that are approximated using the WSM technique are optimized using the Stochastic Gradient Descent technique (Anitha and Vanitha, 2022). The SGD technique is more feasible and has comparatively faster convergence when working with real-time datasets. It works with the logic of updating the model parameters using only a single sample at each iteration. Its inherent noise ensures that the model will not be overfitted and can result in a better generalization by enabling the model to overcome the local minima and saddle points. SDG randomizes the training dataset to mitigate any potential bias caused by the sequence of training samples. The corresponding equation of SGD is shown in Equation (6). The initial weights of the feature are assigned using the WSM approach.

$$\omega' = \omega - \eta \bullet \nabla G(\omega) \quad (6)$$

From the above equation, the notation ω' represents the updated weight, and ω denotes the old weight. The notation η represents the learning rate, and $\nabla G(\omega)$ represents the gradient of the loss function concerning the weights. The value of $G(\omega)$ is determined to using the Equation (7).

$$G(\omega) = \frac{1}{2}(y - y')^2 \quad (7)$$

In the above equation, the notation y designates the actual value, and y' designates the predicted value. The corresponding algorithm for SGD

Table 5
Algorithm for weight optimization using SGD.

Algorithm 1: Weight optimization using SGD.

Input: Weight ω, Learning rate η, Gradient ∇G, Features n

Output: Updated Weight ω'

While true **do**
  for $i = 1,2,3 \ldots, n$ **do**
    Call fun_SGD()
      $\omega' = \omega - \eta \cdot \nabla G(\omega)$
      Call fun_Gradient()
        $G(\omega) = \frac{1}{2}(y - y')^2$
        return $G(\omega)$
    return ω'
  end of **for**
end While

Table 6
Algorithm for weight optimization using ACO.

Algorithm 2: Weight optimization using ACO.

Input: $s \leftarrow number\ of\ iterations$
$n \leftarrow number\ of\ ants$
$\eta \leftarrow hueristic\ function$
$\tau \leftarrow initial\ phenomenon$
$\alpha, \beta \leftarrow weight\ coefficients$
$\nu \leftarrow evaporation\ rate$

Output: Updated weight ω'

Initialize pheromone levels τ_{ij} //assigned based on WSM value
Initialize heuristic values η_{ij}
for i starting from 1 to s
for m starting from 1 to n
Call fun_Probabilistic()

$$\rho_{i,j}^{m}(k) = \frac{[\tau_{ij}(k)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{r \in \Re_i^m} [\tau_{ir}(k)]^{\alpha} \cdot [\eta_{ir}]^{\beta}}$$

Call fun_updatephenomenon()

$$\tau_{ij}(k+1) = (1-\nu) \cdot \tau_{ij}(k) + \sum_{m=1}^{n} \Delta\tau_{ij}^{m}(k)$$

return τ_{ij}
return ω'
end of **for**
end of **for**

is shown in Table 5.

4.3. Feature weight updating using ACO

The second local model is implemented using the Ant Colony optimization technique (Yilmaz Eroglu and Akcan, 2024) for updating the feature weights. Updating feature weights by using the combined behavior of ants to discover optimum solutions in intricate search areas. ACO quickly examines the potential combinations of feature weights via a technique similar to ant pheromone trails, which guides the search to favorable sets. This strategy balances exploration and exploitation dynamically, improving the ability to avoid local optima and reach a more globally optimum set of feature weights (Farghaly et al., 2020). As a result, ACO may increase the precision of classification models by efficiently updating the feature weights. The initial feature weights are obtained from the WSM technique. The objective function concerning the ACO with a pheromone level τ_{ij} for every path between the nodes i and j. The feature weights are selected as a probabilistic path selection among the node concerning the pheromone levels and a heuristic function as shown in Equation (8).

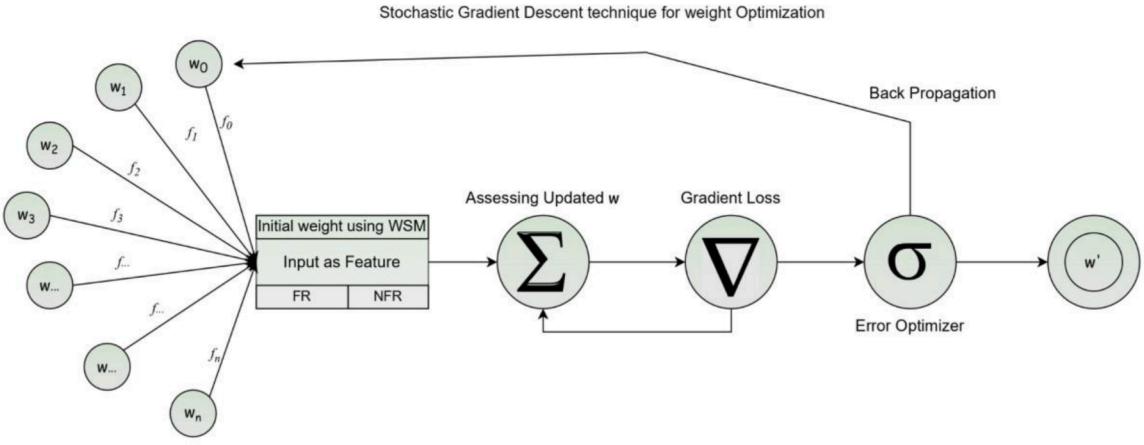

Fig. 7. Feature weight updating mechanism using SGD technique.

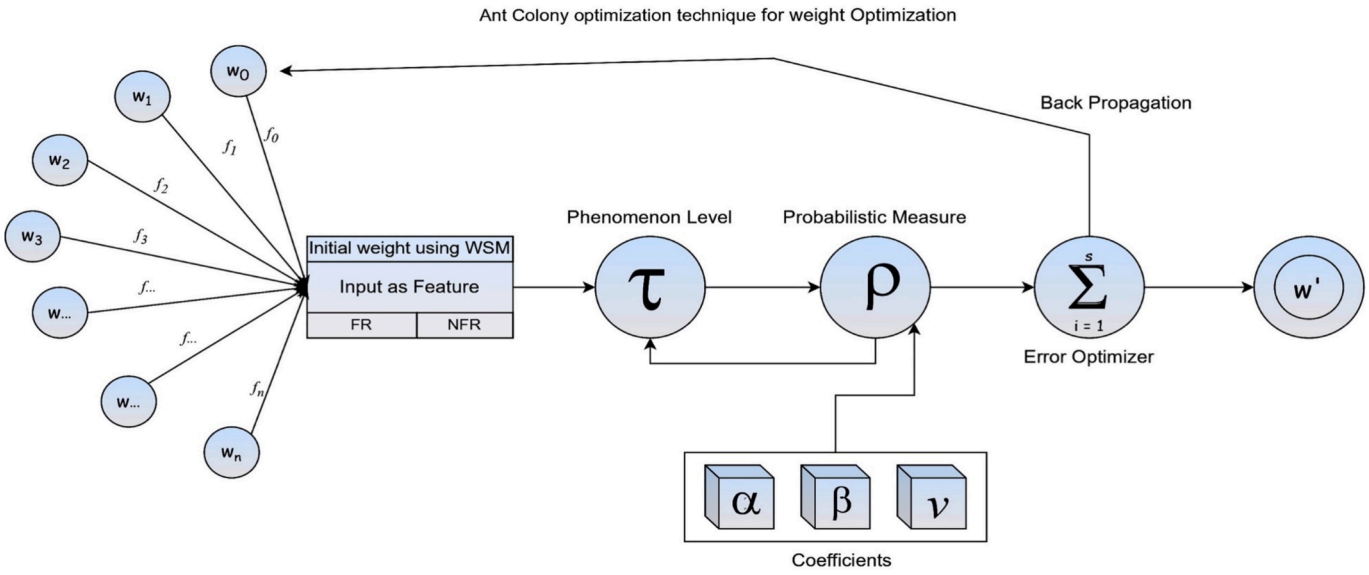

Fig. 8. Feature weight updating mechanism using ACO technique.

$$\rho_{i,j}^{m}(k) = \frac{[\tau_{ij}(k)]^{\alpha} \bullet [\eta_{ij}]^{\beta}}{\sum_{r \in \mathscr{P}_i^m} [\tau_{ir}(k)]^{\alpha} \bullet [\eta_{ir}]^{\beta}} \tag{8}$$

From the above equation, the notation $\tau_{ij}(k)$ denotes the phenomenon level over the path i,j concerning the iteration k. The notation η_{ij} designates the heuristic value and coefficients α, β regulates the pheromone and heuristic information, respectively. The notation $\mathscr{P}_i^m$ designated all possible paths that an ant m can take from a node i. The phenomenon level can be updated using the mathematical function concerning to the evaporation rate ν for n number of ants as shown in Equation (9).

$$\tau_{ij}(k+1) = (1-\nu) \bullet \tau_{ij}(k) + \sum_{m=1}^{n} \Delta\tau_{ij}^{m}(k) \tag{9}$$

From the above equation, the $\Delta\tau_{ij}^{m}(k)$ denotes the magnitude of the displacement of the ant m in the path ij. The corresponding algorithm for SGD is shown in the Table 6.

4.4. Implementation of the local models

The current section discusses the role of local models in updating the feature significance. The features here are the requirements, and the requirement's weight decides the significance of the requirement in the overall prototype, and those requirements are addressed first before the other requirements are addressed. Each of the local models is explained in detail in the current sub-section of the manuscript.

Local Model 1: The first local model is implemented using the Stochastic Gradient Descent technique for updating the feature weights of the model. The initial weights are assigned to the requirement using the WSM. Then, the weights are updated using the SGD technique, where the learning rate and gradient loss are considered when updating the previous weight. SGD randomly selects a single data point for each update, unlike the slower technique that evaluates the full dataset. It calculates the gradient, which shows the direction of weight modification to minimize error. The weights are updated based on the gradient and a learning rate in the opposite direction of the error. This iterative approach cycles over the data through numerous iterations until the model achieves the lowest error. The corresponding architecture diagram is shown in Fig. 7.

Local Model 2: The second local model is implemented using the Ant Colony Optimization technique for updating the initial weights. Artificial "ants" explore a simulated feature space. A path's "fitness" depends on factors like classification performance and feature subset size, which is analogous to a food source's desirability for real ants. Pheromones replicate ant scent trails on selected features. Over time, these trails help ants combine features better. Features' stronger pheromone trails reflect a greater model performance contribution. ACO can efficiently search the feature space and find optimum weight - optimization of the features. The corresponding figure for ACO is shown in Fig. 8.

5. Role of federated learning in DSD

The current section of the manuscript presents the procedure that is followed in aggregating the feature weights to the global model. The feature weights that are assessed at the local models SGD and ACO are aggregated to assess the weight at the global model. Rather than relying on centralized data stored on a single server, federated learning algorithms learns from data distributed across multiple interconnected client devices. Data confidentiality, integrity, security, access rights, and heterogeneity are some of the major issues addressed by a centralized server in most FL implementations, which then helps with training a distributed model. For FL to work, each client securely trains its central model through the model weight rather than sending the actual data (Jiang et al., 2022). Here is a mathematical expression for the FL as shown in Equation (10).

$$\underset{\omega \in R}{Updated} f(\omega, p, q) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{k_i D_{size}} \sum_{j=1}^{k_{i,q} D_{size}} f(\omega_j, p_{sj}, q_{sj}) \tag{10}$$

From the above equation, the three parameters that are given as the input to the weight updating function are the current weight of the data point, i.e., feature. The variables (p, q) represents the coordinates of the matrix that holds the feature weights. The notation n denotes the number of clients in the FL environment. The variable k_i denotes the number of data points in the i^{th} client. The notation D_{size} denotes the size of the data sample associated with the client. The above equation calculates the local mean over all local data samples. The function $f(\omega_j, p_{sj}, q_{sj})$ represents the evaluation of the model function on the j^{th} sample from the i^{th} client, with ω_j being the model parameters and (p_{sj}, q_{sj}) being the j^{th} input–output pair of the s^{th} client. The weight matrix $\omega(p, q)$ was shown in the Equation (11).

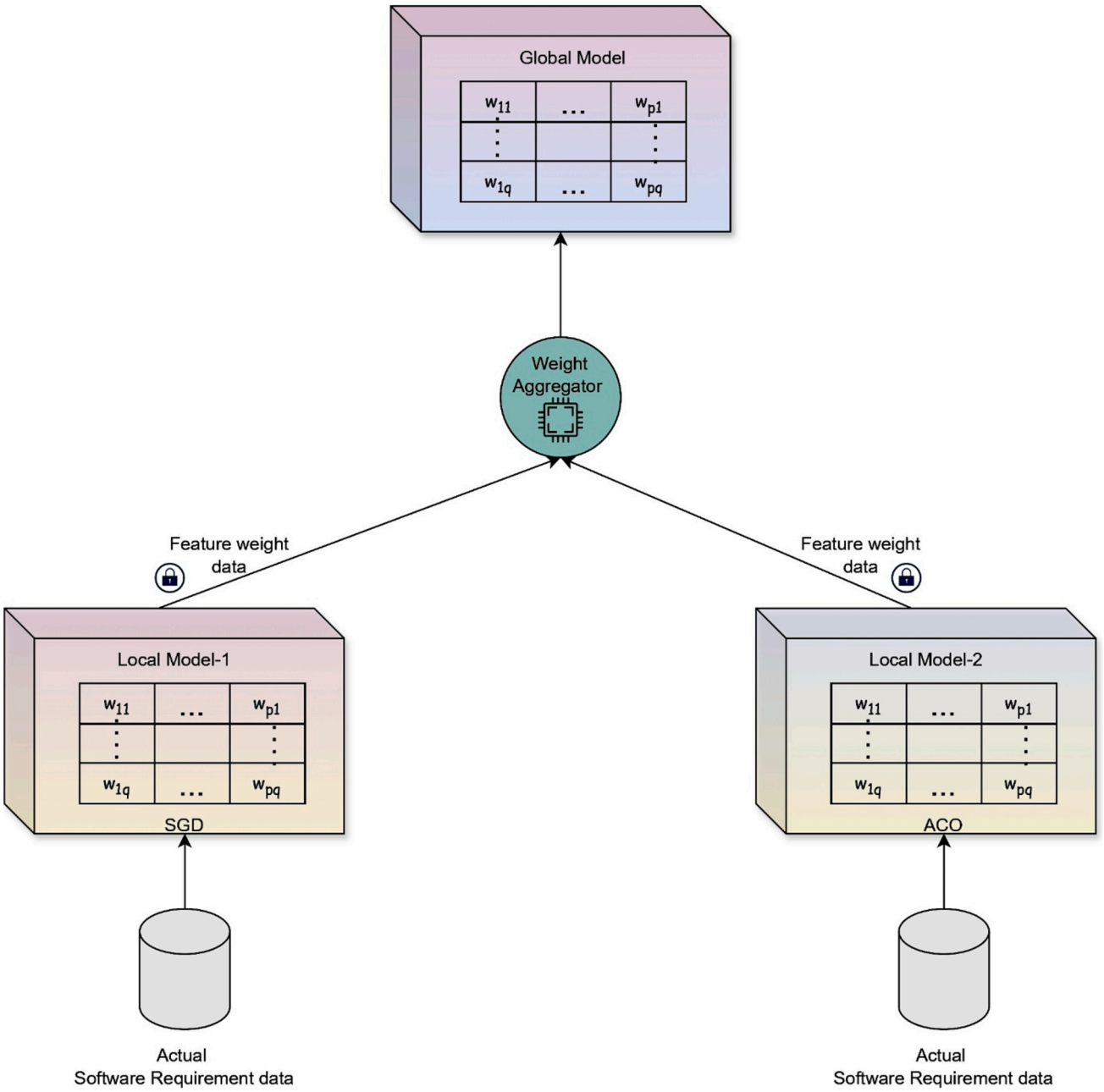

Fig. 9. The FL-based global model is built on local models with a feature aggregator.

$$\omega(p,q) = \begin{bmatrix} \omega_{11} & \cdots & \omega_{1p} \\ \cdot & \cdot & \cdot \\ \omega_{q1} & \cdots & \omega_{pq} \end{bmatrix}_{p \times q} \tag{11}$$

The proposed model is evaluated concerning the loss measure, and it is always desired to have a minimal loss value. The loss is always desired to be minimal for a robust model. The loss measure is regulated by the learning rate associated with the global model. The function for the loss measure is shown in Equation (12).

$$R'_{\omega_i} = \omega_i(t-1) - \gamma \bullet \nabla F_i(\omega_i(t-1)) \tag{12}$$

From the above equation, The symbol ∇ denotes the gradient, which is a vector of partial derivatives concerning the model parameters. The objective function F_i assess the loss associated with the model on the client's local data. The $\omega_i(t-1)$ represents the model parameters pertaining to the i^{th} client at the previous time step. The γ represents the learning rate of the global model. The R'_{ω_i} determined the direction of the gradient. The aggregation of the feature weights is shown in Equation (13)

$$\omega_g = \sum_{i=1}^{n} \frac{\omega_{L1}^i + \omega_{L2}^i}{2} \tag{13}$$

The above equation is evaluated when both the local models have identical features, where the above equation is repeated for all the n features that are considered. The notation ω_{L1}^i designated the weight of the i^{th} feature of local model $L1$ and similarly, ω_{L2}^i designated the weight of the i^{th} feature of local model $L2$. When the feature is associated with only a single local model, the weight was assessed, as shown in Equation (14).

$$\omega_g = \sum_{i=1}^{n} \frac{\omega_x^i + \omega_x^i}{2} \tag{14}$$

From the above equation, the notation ω_x^i denotes the i^{th} feature

Table 7
The loss measures associated with the function requirements of the E-commerce website.

Requirements	MSE	MAE	PCC
FR1:User Account Creation	0.012	0.134	0.998
FR2:User Authentication	0.087	0.173	0.996
FR3:User Profile Management	0.203	0.338	0.982
FR4: Product Catalog	0.105	0.293	0.987
FR5: Product Reviews	0.304	0.384	0.981
FR6: Shopping Basket	0.110	0.200	0.986
FR7: Order Management	0.181	0.314	0.987
FR8: Payment Processing	0.027	0.192	0.993

Table 8
The loss measures are associated with the function requirements of the E-commerce website.

Requirements	MSE	MAE	PCC
NFR1: Data Privacy	0.207	0.332	0.964
NFR2: Scalability	0.146	0.227	0.981
NFR3: Maintainability	0.111	0.198	0.990
NFR4: Cross-Browser Compatibility	0.224	0.305	0.952
NFR5: Load Handling	0.173	0.294	0.989
NFR6: Interoperability	0.096	0.21	0.973
NFR7: Audibility	0.237	0.289	0.968
NFR8: Version Control	0.154	0.318	0.987

weight of the local model x. Where the same weight is assigned to the global model in case the feature is associated with a single local model. The corresponding figure for FL with two local models, as discussed in the current study, is shown in Fig. 9.

The FL model can be further incorporated with multiple local models for requirement prioritization. Adding more local models would result in a considerable trade-off between the accuracy and the computational resources needed to maintain the network. However, the current study is confined to two local models for building the FL network.

6. Results and discussion

The performance of the proposed FL model for DSD is evaluated, and various loss measures like MSE, MAE, and PCC are used to evaluate the performance of the model. The feature weights determine the significance of the requirements, the loss measure illustrates the actual weight that is associated with the requirement and the assessed weight using the FL model from the local models. It is desired to have a minimal loss when evaluating the difference between the predicted loss and the actual loss. The loss associated with the local model and the global model is being assessed for some of the FR and NFR that are considered in the current study. The evaluations are done on both the case studies that are considered in the current study.

6.1. Observation of case study: E-commerce website

The requirements associated with the e-commerce website prototype are evaluated in relation to various loss measures. The FR and NFR are separately evaluated to assess the robustness of the model. Only some of the requirements from both FR and NFR are considered to confine the study. The evaluation of the FR is shown in Table 7 and the NFR evaluations are tabulated in Table 8. Figs. 10 and 11 represents the graphs associated with loss measures for FR and NFR, respectively.

It can be observed from the above graphs that the loss associated with the FR is comparatively lesser than the NFR. This could be due to diversity and dependencies associated with the requirements.

6.2. Observation of case study: Learning Management System

The performance of the proposed model for evaluating in developing the Learning Management System prototype. The FR and NFR are evaluated using the loss measures to determine the robustness of the model. Some of the FR and NFR are in common for both case studies. The corresponding loss values observed when evaluating the FR are presented in Table 9, and the corresponding graphs are shown in Fig. 12. The corresponding loss values associated with NFR are presented in Table 10, and their corresponding graphs are in Fig. 13.

It can be observed from the loss measures that the FR has comparatively lesser loss values than the NFR. The NFR priorities would change for each software that is designed and the development model that is used in building the prototype.

A table that summarizes the performances of the proposed model across two distinct case studies for both FR and NFR of eight each is shown in Table 11. The statistical analysis would assist in better analyzing the performance of the DSD model in the FL environment.

It can be observed from the above table, across both the case studies,

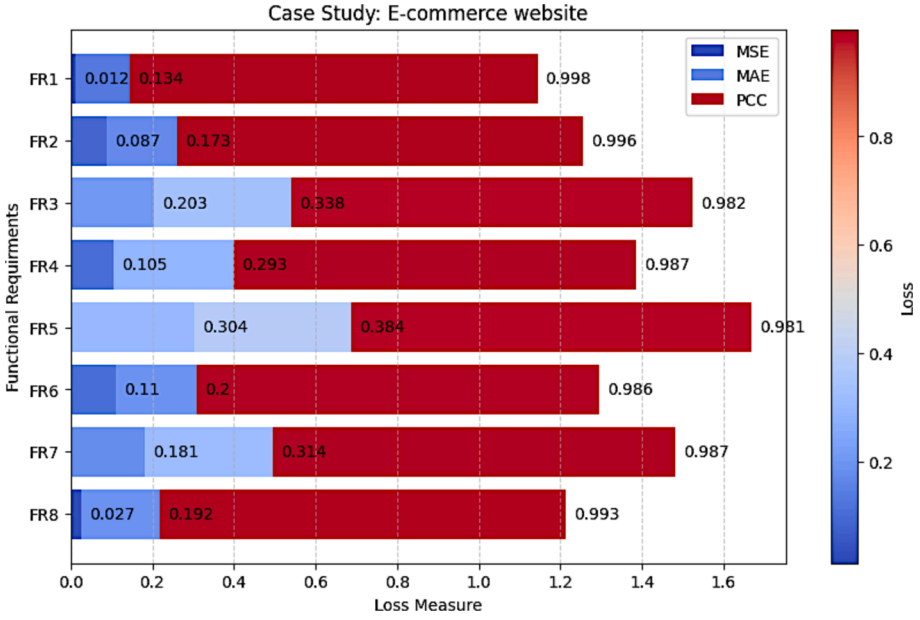

Fig. 10. The graph represents the loss measure associated with the Functional Requirements in E-commerce website.

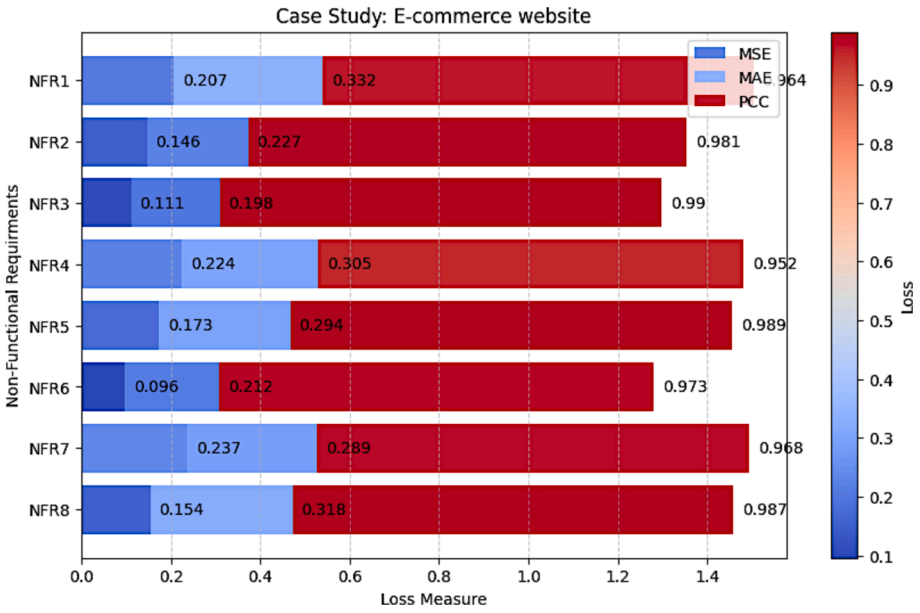

Fig. 11. The graph represents the loss measure associated with the Non-Functional Requirements in E-commerce website.

Table 9
The loss measures are associated with the functional requirements of the Learning Management System prototype.

Requirements	MSE	MAE	PCC
FR1:User Account Creation	0.017	0.102	0.997
FR2:User Authentication	0.048	0.150	0.996
FR3:User Profile Management	0.189	0.287	0.990
FR4: Course Management	0.093	0.119	0.986
FR5: Teacher-Course Mapping	0.185	0.221	0.984
FR6: Student-Course Mapping	0.176	0.207	0.982
FR7: Course Organizer	0.212	0.257	0.968
FR8: Assessment Module	0.138	0.186	0.990

that the FR has lesser loss compared to NFR, as FR is given more significance as they are the immediate requirements that the client would assess. Hence, the FR is implemented prior to that giving weightage to the NFR. Furthermore, the time delay associated with the combined implementation of the proposed model for both FR and NFR for both case studies is shown in Fig. 14. The time delay presented in the above

Table 10
The loss measures are associated with the function requirements of the Learning Management System prototype.

Requirements	MSE	MAE	PCC
NFR1: Responsiveness	0.198	0.243	0.977
NFR2: Scalability	0.117	0.132	0.990
NFR3: Maintainability	0.099	0.126	0.993
NFR4: Latency	0.099	0.171	0.990
NFR5: Customization	0.191	0.224	0.969
NFR6: Interoperability	0.105	0.200	0.981
NFR7: Modularity	0.162	0.246	0.975
NFR8: Documentation	0.143	0.212	0.988

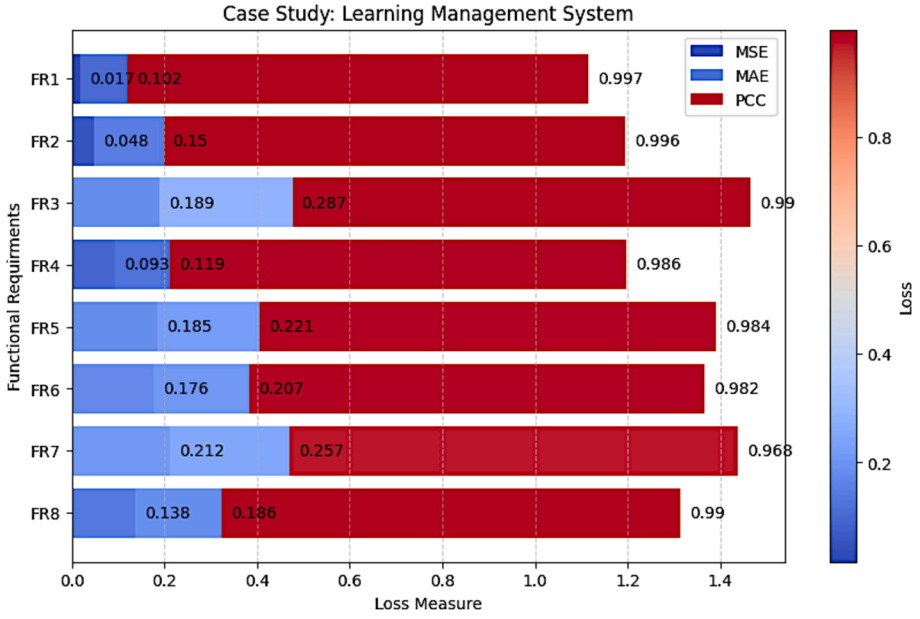

Fig. 12. The graph represents the loss measure associated with the Functional Requirements in the Learning Management System prototype.

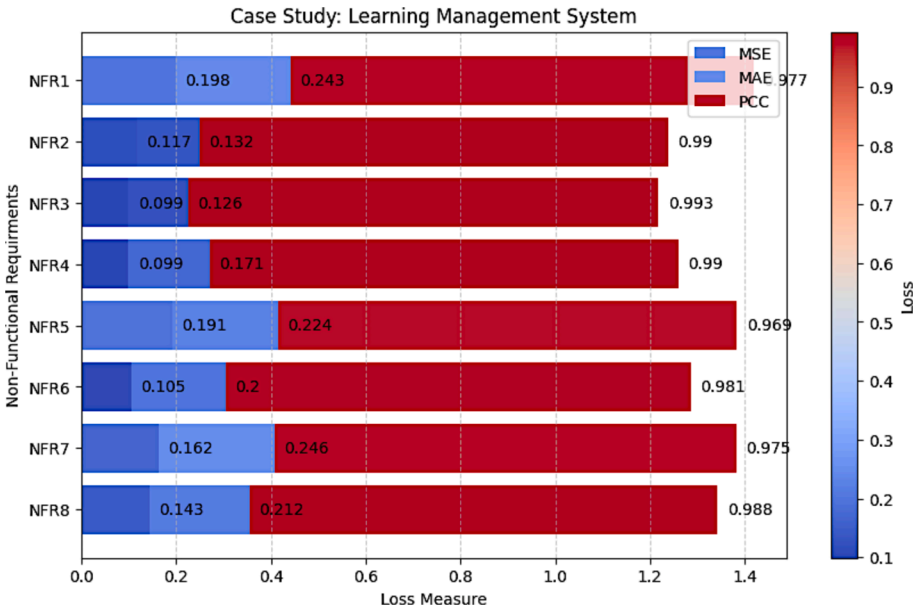

Fig. 13. The graph represents the loss measure associated with the Non-Functional Requirements in the Learning Management System prototype.

Table 11
The summary of performances of the DSD model in the FL Environment.

Case Study	Type of Requirement	MSE	MAE	PCC
E-commerce website	FR	0.128	0.253	0.988
	NFR	0.168	0.271	0.975
Learning Management System	FR	0.132	0.191	0.986
	NFR	0.139	0.194	0.982

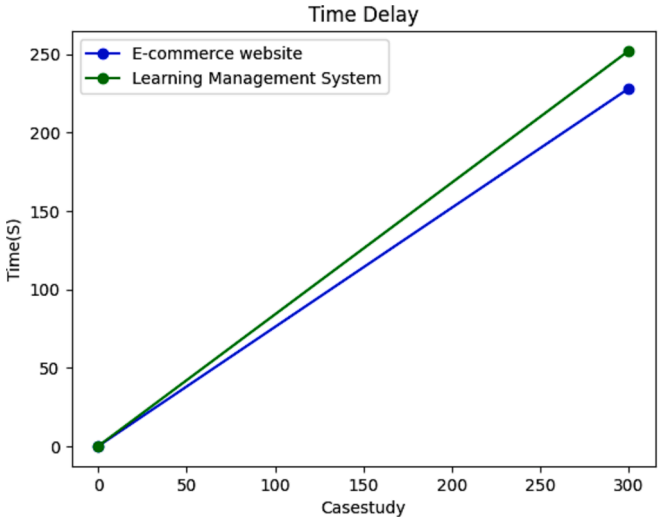

Fig. 14. The graph represents the time delay associated with each of the case studies.

figure is only associated with feature weight engineering, which includes the initial weight assessment and updating the weight. The time delay is shown in the total time for both the FR and NFR in both the test cases. The time delay largely depends on the underlying model used in weight updating, the number of features being considered, and the hyperparameters associated with the implementation.

6.3. Latency analysis

The latency measure indicated the total delay incurred in implementing the FL model through the local models. The delay is associated with pushing and aggregating the local weights at the global model. Many factors influence the latency of the network delay, processing delay, computational delay at the aggregator, and the number of features being processed (Ahmed et al., 2023). The delay associated with implementing each case study over multiple iterations is presented in Table 11, and the corresponding graphs are shown in Fig. 15.

The delay is largely associated with the number of features associated with each software prototype. The latency increases with the number of features and iterations associated with evaluation. The details of the time delay must be further evaluated concerning network delay, processing delay, queuing delay, and delay associated with updating the value at the global model for better comprehensiveness of the model.

6.4. Practical implications

In the federated learning environment, the global model is built at the top of multiple local models, i.e., the feature weights at the local level are fed as input for the aggregator at the global model. When divergent local models are being implemented over distinct projects, the NFR for each project would vary, a particular NFR might be of higher priority for a particular project, and it may be least significant to the other. In such cases, the aggregation might inappropriately assess the feature weight for the global model. This is considered one of the major of the FL base DSD mechanism. The technical challenges include data heterogeneity across divergent software prototype models and Feature Inconsistencies. The feature significance largely depends on purpose and requirement elucidation by the client concerning the prototype. Inappropriate aggregation of feature weights would mislead the feature significance in the global model. Some of the other challenges associated with FL are communication overhead, latency issues, and coordinating the updates from various nodes at the global model synchronously. These are a few challenges that can be focused on in future research in FL-based DSD approaches.

The FL endorses the security of the information shared over the web, where the feature weights are shared on the internet, rather than the feature-related information. The current model did not analyze the security aspects of the FL environment. The security and privacy of the

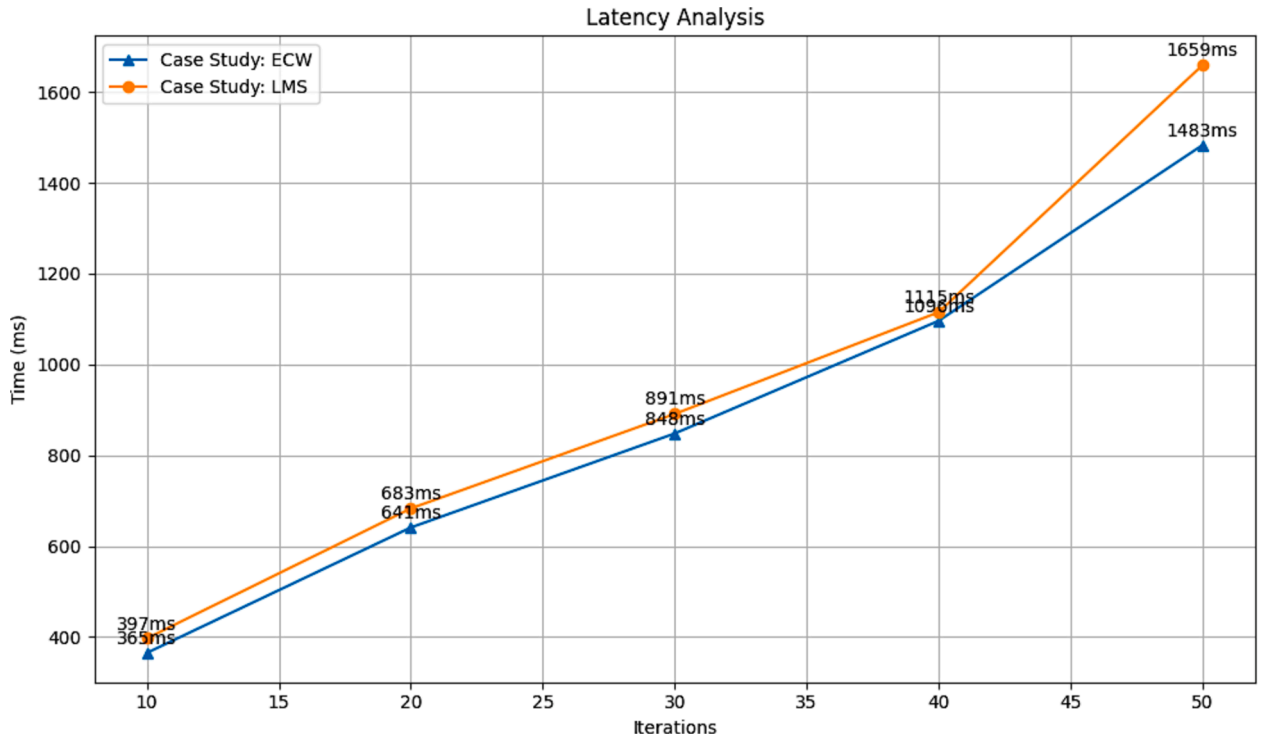

Fig. 15. The graph represents the latency associated with each case study over multiple iterations.

data are the pivotal point of federated learning, which must be further analyzed for better comprehensibility of the study. The current study's metrics evaluated concerning federated learning are confined to latency. However, the privacy concerns are not evaluated, which is the potential limitation of the current study. The privacy of the information and delays associated with model building may be further evaluated for better statistical analysis of the model.

7. Conclusion

The current study on the federated learning-based distributed software development model portrays the feasibility of implementing the rapid prototyping of the agile software development model and the practical implications thereof. The current study has considered two local models to work with the global model, where the feature weights assessed at the local models using the Stochastic Gradient Descent technique and Ant Colony Optimization techniques are aggregated together to formulate the weights at the global model. The study is evaluated across a range of functional and non-functional requirements for both the case studies on the E-Commerce website and Learning management system prototypes, and the performances are assessed using loss measures like MSE, MAE, and Pearson Correlation Coefficient. It is observed that the model has exhibited reasonable performance in terms of evaluation metrics. The latency associated with processing the requirements over multiple iterations is evaluated in the current study.

The model can be further evaluated by examining more features and divergent evaluation metrics. The federated learning model for distributed software development can be further evaluated by implementing the local model over two distinct software prototypes and integrating them over the global model distinct from the local models, which would be the more appropriate way of evaluating the federated learning environment. The performance of the model federated learning model may be further evaluated using divergent evaluation metrics and with more local models to analyze the suitability of the software engineering and requirement engineering domains. Robust model convergence may be achieved by investigating improved aggregation methods and adaptive learning strategies, which can handle non-independently and identically distributed data. Managing versions of FL models and maintaining integrity across remote models is exceptionally significant in developing a robust software model. This is considered to be one of the exceptionally important concerns in the FL environment, and it is considered to be one of the significant future research directions.

Author Contributions

A.A. and S.A. acquired the data for experimentation and drafted the initial draft. S.A. conceptualized the study. S.A. and A.A. investigated and analyzed the results. A.A. has administrated the project. A.A. and S.A. reviewed and edited the document. Both authors have read and agreed to the published version of the manuscript.

Funding

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia [Grant No. KFU241945].

Ethical Approval

The authors declare that they have no ethical approval required for this article.

CRediT authorship contribution statement

Abdulaziz Alhumam: Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization. **Shakeel Ahmed:** Writing – review & editing, Writing – original draft, Validation, Investigation, Conceptualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References:

Ahmed, S., Naga Srinivasu, P., Alhumam, A., 2023. A Software Framework for Intelligent Security Measures Regarding Sensor Data in the Context of Ambient Assisted Technology. Sensors 23, 6564. https://doi.org/10.3390/s23146564.

Alharbi, A., Ansari, M.T.J., Alosaimi, W., Alyami, H., Alshammari, M., Agrawal, A., Kumar, R., Pandey, D., Khan, R.A., 2022. An Empirical Investigation to Understand the Issues of Distributed Software Testing amid COVID-19 Pandemic. Processes 10, 838. https://doi.org/10.3390/pr10050838.

Alshehri, M.S., Saidani, O., Alrayes, F.S., Abbasi, S.F., Ahmad, J., 2024. A Self-Attention-Based Deep Convolutional Neural Networks for IIoT Networks Intrusion Detection. IEEE Access 12, 45762–45772. https://doi.org/10.1109/ACCESS.2024.3380816.

Anitha, S., Vanitha, M., 2022. A novel feature selection with stochastic gradient descent logistic regression for multilabeled stress prediction in working employees. Concurrency Computat Pract Exper. 34 (13), e6911.

C. B. Barona and M. R. Ramirez, "Effects of COVID 19 lockdown on the use of LMS platforms for virtual education," 2021 16th Iberian Conference on Information Systems and Technologies (CISTI), Chaves, Portugal, 2021, pp. 1-6, doi: 10.23919/CISTI52073.2021.9476645.

Costa, P., Rodrigues, H., 2023. The ever-changing business of e-commerce-net benefits while designing a new platform for small companies. Rev Manag Sci. https://doi.org/10.1007/s11846-023-00681-6.

Y. Cui, K. Cao, J. Zhou, and T. Wei, "HELCFL: High-Efficiency and Low-Cost Federated Learning in Heterogeneous Mobile-Edge Computing," *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Antwerp, Belgium, 2022, pp. 1227-1232, doi: 10.23919/DATE54114.2022.9774662.

David, J., Thaís, B., Everton, C., 2023. An Agile Management Model for Distributed Software Development Teams, SBES. XXXVII Brazilian Symposium on Software Engineering 2023, 122–131. https://doi.org/10.1145/3613372.3613401.

Domingos Alves, D., Souza Matos, E. de, & Chavez, C. von F. G. (2023). Interaction design in distributed software development: a systematic mapping study. Behaviour & Information Technology, 1–37. Doi: 10.1080/0144929X.2023.2286534.

El Koshiry, A.M., Eliwa, E.H.I., Abd El-Hafeez, T., Khairy, M., 2024. Detecting cyberbullying using deep learning techniques: a pre-trained glove and focal loss technique. PeerJ Computer Science 10, e1961.

Emmanuel, T., Maupong, T., Mpoeleng, D., et al., 2021. A survey on missing data in machine learning. J Big Data 8, 140. https://doi.org/10.1186/s40537-021-00516-9.

Farghaly, H.M., Ali, A.A., El-Hafeez, T.A. (2020). Developing an Efficient Method for Automatic Threshold Detection Based on Hybrid Feature Selection Approach. In: Silhavy, R. (eds) Artificial Intelligence and Bioinspired Computational Methods. CSOC 2020. Advances in Intelligent Systems and Computing, vol 1225. Springer, Cham. Doi: 10.1007/978-3-030-51971-1_5.

Farooq, M.S., Kalim, Z., Qureshi, J.N., Rasheed, S., Abid, A., 2022. A Blockchain-Based Framework for Distributed Agile Software Development. IEEE Access 10, 17977–17995. https://doi.org/10.1109/ACCESS.2022.3146953.

Marum Simão Filho, Plácido R. Pinheiro, Adriano B. Albuquerque, Joel J. P. C. Rodrigues, "Task Allocation in Distributed Software Development: A Systematic Literature Review", Complexity, vol. 2018, Article ID 6071718, 13 pages, 2018. Doi: 10.1155/2018/6071718.

Guleria, P., Naga Srinivasu, P., Ahmed, S., Almusallam, N., Alarfaj, F.K., 2022. XAI Framework for Cardiovascular Disease Prediction Using Classification Techniques. Electronics 11, 4086. https://doi.org/10.3390/electronics11244086.

K. B. Ijaz, I. Inayat and F. Allah Bukhsh, "Non-functional Requirements Prioritization: A Systematic Literature Review," 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Kallithea, Greece, 2019, pp. 379-386, doi: 10.1109/SEAA.2019.00064.

Jabangwe, R., Šmite, D., Hessbo, E., 2016. Distributed software development in an offshore outsourcing project: A case study of source code evolution and quality. Information and Software Technology 72, 125–136. https://doi.org/10.1016/j.infsof.2015.12.005.

Jadhav and R. Sonar, "Analytic Hierarchy Process (AHP), Weighted Scoring Method (WSM), and Hybrid Knowledge Based System (HKBS) for Software Selection: A Comparative Study," 2009 Second International Conference on Emerging Trends in Engineering & Technology, Nagpur, India, 2009, pp. 991-997, doi: 10.1109/ICETET.2009.33.

L. Jiang, H. Zheng, H. Tian, S. Xie, and Y. Zhang, "Cooperative Federated Learning and Model Update Verification in Blockchain-Empowered Digital Twin Edge Networks," in IEEE Internet of Things Journal, vol. 9, no. 13, pp. 11154-11167, 1 July 2022, doi: 10.1109/JIOT.2021.3126207.

A. F. Khan et al., "Towards cost-effective and resource-aware aggregation at Edge for Federated Learning," 2023 IEEE International Conference on Big Data (BigData), Sorrento, Italy, 2023, pp. 690-699, doi: 10.1109/BigData59044.2023.10386691.

L'Erario, A., Gonçalves, J., Fabri, J.A., et al., 2020. CFDSD: a Communication Framework for Distributed Software Development. J Braz Comput Soc 26, 7. https://doi.org/10.1186/s13173-020-00101-7.

Le, J., Zhang, D., Lei, X., Jiao, L., Zeng, K., Liao, X., 2023. Privacy-Preserving Federated Learning With Malicious Clients and Honest-but-Curious Servers. IEEE Transactions on Information Forensics and Security 18, 4329–4344. https://doi.org/10.1109/TIFS.2023.3295949.

Li, G., Zhang, A., Zhang, Q., Wu, D., Zhan, C., May 2022. Pearson Correlation Coefficient-Based Performance Enhancement of Broad Learning System for Stock Price Prediction. IEEE Transactions on Circuits and Systems II: Express Briefs 69 (5), 2413–2417. https://doi.org/10.1109/TCSII.2022.3160266.

Mamdouh Farghaly, H., Abd, E.-H., 2022. A new feature selection method based on frequent and associated itemsets for text classification. Concurrency Computat Pract Exper. 34 (25), e7258.

Mamdouh Farghaly, H., Abd El-Hafeez, T., 2023. A high-quality feature selection method based on frequent and correlated items for text classification. Soft Comput 27, 11259–11274. https://doi.org/10.1007/s00500-023-08587-x.

S. S. M. Monfared and A. Kamandi, "Agile techniques and frameworks based on the requirements for e-commerce applications," 2016 Second International Conference on Web Research (ICWR), Tehran, Iran, 2016, pp. 131-138, doi: 10.1109/ICWR.2016.7498457.

Monsalve-Obreque, P., Vargas-Villarroel, P., Hormazabal-Astorga, Y., Hochstetter-Diez, J., Bustos-Gómez, J., Diéguez-Rebolledo, M., 2023. Proposal to Improve the E-Commerce Platform Development Process with an Exploratory Case Study in Chile. Appl. Sci. 13, 8362. https://doi.org/10.3390/app13148362.

Mostafa, G., Mahmoud, H., Abd El-Hafeez, T., et al., 2024. Feature reduction for hepatocellular carcinoma prediction using machine learning algorithms. J Big Data 11, 88. https://doi.org/10.1186/s40537-024-00944-3.

Qi, P., Chiaro, D., Guzzo, A., Ianni, M., Fortino, G., Piccialli, F., 2024. Model aggregation techniques in federated learning: A comprehensive survey. Future Generation Computer Systems 150, 272–293. https://doi.org/10.1016/j.future.2023.09.008.

Qin, Y., Liu, H., 2022. Application of Value Stream Mapping in E-Commerce: A Case Study on an Amazon Retailer. Sustainability 14, 713.

Raj Kumar Chopra, 2016. Varun Gupta, Durg Singh Chauhan, Experimentation on accuracy of non functional requirement prioritization approaches for different complexity projects. Perspectives in Science 8, 79–82. https://doi.org/10.1016/j.pisc.2016.04.001.

Shah, T., 2016. SV Patel, A Novel Approach for Specifying Functional and Non-functional Requirements Using RDS (Requirement Description Schema). Procedia Computer Science 79, 852–860. https://doi.org/10.1016/j.procs.2016.03.083.

Shankar, P., Morkos, B., Yadav, D., et al., 2020. Towards the formalization of non-functional requirements in conceptual design. Res Eng Design 31, 449–469. https://doi.org/10.1007/s00163-020-00345-6.

Siddiqa, H.A., et al., 2024. Single-Channel EEG Data Analysis Using a Multi-Branch CNN for Neonatal Sleep Staging. IEEE Access 12, 29910–29925. https://doi.org/10.1109/ACCESS.2024.3365570.

Siddique, A.A., Raza, A., Alshehri, M.S., Alasbali, N., Abbasi, S.F., 2024. Optimizing Tumor Classification Through Transfer Learning and Particle Swarm Optimization-Driven Feature Extraction. IEEE Access 12, 85929–85939. https://doi.org/10.1109/ACCESS.2024.3412412.

Taweel, A., Brereton, P., 2006. Modelling software development across time zones. Information and Software Technology 48 (1), 1–11. https://doi.org/10.1016/j.infsof.2004.02.006.

Wen, J., Zhang, Z., Lan, Y., et al., 2023. A survey on federated learning: challenges and applications. Int. J. Mach. Learn. & Cyber. 14, 513–535. https://doi.org/10.1007/s13042-022-01647-y.

Z. Xu et al., "HierFedML: Aggregator Placement and UE Assignment for Hierarchical Federated Learning in Mobile Edge Computing," in IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 1, pp. 328-345, 1 Jan. 2023, doi: 10.1109/TPDS.2022.3218807.

Yang, Y., Xing, Hu., Gao, Z., Chen, J., Ni, C., Xia, X., Lo, D., 2024. Federated Learning for Software Engineering: A Case Study of Code Clone Detection and Defect Prediction. IEEE Transactions on Software Engineering 50 (02), 296–321. https://doi.org/10.1109/TSE.2023.3347898.

Yilmaz Eroglu, D., Akcan, U., 2024. An Adapted Ant Colony Optimization for Feature Selection. Applied Artificial Intelligence 38 (1). https://doi.org/10.1080/08839514.2024.2335098.

Younas, M., Jawawi, D.N.A., Mahmood, A.K., Ahmad, M.N., Sarwar, M.U., Idris, M.Y., 2020. Agile Software Development Using Cloud Computing: A Case Study. IEEE Access 8, 4475–4484. https://doi.org/10.1109/ACCESS.2019.2962257.