future internet

MDPI

Article

Optimizing Requirements Prioritization for IoT Applications Using Extended Analytical Hierarchical Process and an Advanced Grouping Framework

Sarah Kaleem [1,2], Muhammad Asim [1,3,*], Mohammed El-Affendi [1] and Muhammad Babar [4,*]

[1] EIAS Data Science Lab, Prince Sultan University, Riyadh 11586, Saudi Arabia; skaleem@psu.edu.sa or sarahkaleem33887@iqraisb.edu.pk (S.K.); affendi@psu.edu.sa (M.E.-A.)
[2] Department of Computing and Technology, Iqra University, Islamabad 44000, Pakistan
[3] School of Computer Science and Technology, Guangdong University of Technology, Guangzhou 510006, China
[4] Robotics and Internet of Things Lab, Prince Sultan University, Riyadh 11586, Saudi Arabia
* Correspondence: masim@psu.edu.sa (M.A.); mbabar@psu.edu.sa (M.B.)

Abstract: Effective requirement collection and prioritization are paramount within the inherently distributed nature of the Internet of Things (IoT) application. Current methods typically categorize IoT application requirements subjectively into inessential, desirable, and mandatory groups. This often leads to prioritization challenges, especially when dealing with requirements of equal importance and when the number of requirements grows. This increases the complexity of the Analytical Hierarchical Process (AHP) to O(n2) dimensions. This research introduces a novel framework that integrates an enhanced AHP with an advanced grouping model to address these issues. This integrated approach mitigates the subjectivity found in traditional grouping methods and efficiently manages larger sets of requirements. The framework consists of two main modules: the Pre-processing Module and the Prioritization Module. The latter includes three units: the Grouping Processing Unit (GPU) for initial classification using a new grouping approach, the Review Processing Unit (RPU) for post-grouping assessment, and the AHP Processing Unit (APU) for final prioritization. This framework is evaluated through a detailed case study, demonstrating its ability to effectively streamline requirement prioritization in IoT applications, thereby enhancing design quality and operational efficiency.

Keywords: requirement engineering; internet of things; requirements prioritization; analytical hierarchical process (AHP)

Citation: Kaleem, S.; Asim, M.; El-Affendi, M.; Babar, M. Optimizing Requirements Prioritization for IoT Applications Using Extended Analytical Hierarchical Process and an Advanced Grouping Framework. *Future Internet* **2024**, *16*, 160. https://doi.org/10.3390/fi16050160

Academic Editor: Ping Wang

Received: 19 March 2024
Revised: 17 April 2024
Accepted: 29 April 2024
Published: 6 May 2024

1. Introduction

Collecting and prioritizing requirements in Internet of Things (IoT) applications is a critical and challenging task due to their distributed nature. Selecting the correct requirements is essential for designing high-quality IoT systems. Requirements in such systems are often interdependent and vary significantly across different systems, influenced by the attributes of the system, the stakeholders involved, and the operational context [1]. Consequently, requirement engineering (RE) methods need to be tailored to the specific characteristics of each system. IoT applications, in particular, present complex challenges in requirement management due to their conflicting and interdependent nature [2], which complicates their prioritization. Contradiction, interference, and conflict among applications make the nature of nature more complex. This complex nature is due to the interdependencies that need to be prioritized. Understanding interdependencies is very important among requirements (REQs) in prioritizing REQs. Interdependency not only causes prioritization but also causes cooperation among REQs. If proper prioritization is not carried out, it will cause the dismissal of the requirement or lead a project to failure

in later phases. So the main hurdle in developing quality software is dealing with the prioritization of REQs.

Prioritization of requirements plays a significant role in decision-making. A set is nominated, based on its significance, using prioritization schemes. Choosing first the most essential requirements to implement becomes vital when there are vast requirements [3]. Prioritization is crucial for identifying the most critical tasks for an initial prototype. It is necessary to select the correct requirements for quality-oriented results [4]. It is further argued that selecting the correct requirement directly affects the quality [5]. The critical requirements must be prioritized, when there are limitations on resources and time, to ensure a project's success [6]. Choosing the correct requirements is a significant decision. Prioritization methods can be classified into two classes: negotiation and systematic methods. Various aspects (attribute or property) are considered while prioritizing [7,8]. The preferences are usually based on schedule, budget, cost, or importance. The Business Value considers various requirements that impact the overall production when executed. A few requirements might be essential, others might be average, and a few might be unnecessary, like wish lists and gold plating. The desirability of a requirement is directly proportional to the value it offers.

The factors influencing project cost include code reusability, the complexity of requirements, documentation, and testing. Cost evaluation is typically measured in hours required to complete specific functionalities. The desirability of requirements is directly proportional to the value they provide [9]. Additionally, every project faces various uncertainties or risks that necessitate effective risk management strategies in order to address both external and internal challenges. It is essential to prioritize requirements while also considering the associated risks. Desirability is also directly related to cost. The time it takes to bring a product to market is significantly influenced by the availability of the resources necessary for job completion. Factors such as concurrent development, the need for training, and adherence to industry standards also impact time-to-market. The instability of requirements is another critical factor; requirements often change, particularly during the development phase, due to shifts in the market, government regulations, or industry standards [10]. Therefore, it is advisable to first execute the more stable requirements. The debate highlights that while various dimensions are significant, not every dimension needs to be considered in the prioritization process. The selection of what to prioritize is contingent on the specific situation and circumstances [11].

This article introduces a comprehensive framework integrating the Analytical Hierarchical Process (AHP) and an advanced grouping methodology to address the deficiencies of current prioritization practices in IoT-enabled edge solutions. This framework not only refines existing approaches to mitigate the limitations and subjectivity of traditional grouping models but also expands the capacity of AHP to handle a broader range of requirements efficiently. The framework is organized into two main modules: the Pre-processing Module, where requirements are collected and stored in a repository, including real-time requirements, and the Prioritization Module, which is tasked with grouping, reviewing, and prioritizing these requirements. Within the Prioritization Module, the Grouping Processing Unit (GPU) classifies requirements using a newly proposed grouping approach; the Review Processing Unit (RPU) assesses the requirements post-grouping; and the AHP Processing Unit (APU) is responsible for the final prioritization of the requirements.

The remainder of the article is structured as follows: Section 2 provides a literature review that critiques existing methods and identifies the gaps that our framework addresses. Section 3 describes the proposed framework in detail, explaining the workings and components of each module. Section 4 discusses the validation and conducts a case study to demonstrate the framework's practical applications and effectiveness. Section 5 sets out the challenges and limitations associated with implementing the framework. Finally, Section 6 concludes the article with a summary of our findings.

2. Literature Review

Requirements prioritization is a significant step in requirements collection, particularly in the applications of IoT and smart cities. Several approaches can be found in the literature for prioritization of requirements. In IoT, the requirements are frequently distributed over different stakeholders and various organizations [12]. The list of significant problems with the current techniques comprises less accuracy, low quantitative measurement, requirements for prioritization time, and low maturity scalability [13–15]. To design suitable IoT-enabled applications, it is essential to choose correct REQs [16,17]. In IoT-enabled applications, there might be a considerable number of distributed stakeholders, which causes difficulties in requirements management. Requirements in IoT are prioritized using tools, e.g., forums or wikis, for the use of which teamwork among the developers and customers is essential [18]. The user can also prioritize the REQs using various methods. Many prioritization methods are used based on the application. There are four scales for prioritization: ordinal scale, nominal scale, ratio scale, and interval scale [18]. Some broadly used methods are discussed in this section.

The Analytical Hierarchical Process (AHP) is a systematic and prominent method used for allocating priorities and making accurate judgments within large sets of requirements. It involves a pairwise comparison between two REQs to determine which has the higher priority, using a ratio scale [5]. The number of necessary comparisons is represented by $n(n-1)/2$ at each hierarchical level, escalating as the number of requirements increases to O(n^2), which is recognized as a significant limitation of AHP, although it remains a dependable method for handling a moderate number of requirements [19,20]. Another method, known as Cumulative Voting, or the 100-dollar trial [19], involves distributing 100 fictitious votes among the members, who then allocate these votes to the REQs based on their perceived importance. The votes for each REQ are then totaled to determine the priorities [21]. However, this approach has drawbacks, such as potential bias when a member allocates all of their votes to a single REQ personally deemed important but not determined as such for others, and it is less effective when the number of REQs is very large (e.g., REQs ≥ 100), possibly leading to inaccuracies in prioritization.

The Grouping method is also called numerical assignment. It is a very well-known and straightforward prioritization method. It is based on an ordinal scale [22]. Generally, three groups are formed (e.g., optional, standard, and critical). A central tactic in this method is dividing REQs into only three classes, as every stakeholder considers the degree to which their requirements are necessary. The management of class requirements of exact priority is another issue. In the ranking method, the requirements are prioritized; the most significant REQ is ranked as number 1, the second most significant is ranked as number 2, the second most significant is ranked as number 3, and so forth. Accordingly, the number n is of the slightest significance. This mechanism utilizes an ordinal scale. There is not an obvious way to check relative variation among 2 REQs, in the manner of AHP [19–21]. It is suitable when only one stakeholder prioritizes all of the requirements, and it is used for small projects. Top-ten requirements handling is the most straightforward technique in terms of sophistication and roughness of granularity [23]. Here, the top 10 requirements are selected from the vast set, without any number assignment, and it is not based on any scale. It is helpful when multiple users have the same assessment. The disadvantage is that there is no ordered assignment for the top 10 requirements, which makes decision-making during implementation very challenging. The Binary Search Tree (BST) is another specific method for prioritization. Every object is a particular requirement in BST. The REQs sorted in the leftmost sub-tree are less important, while REQs on the rightmost sub-tree are more crucial; the root node is not moderate [23]. This method limits the number of comparisons in AHP by dividing the REQs into two sub-trees from every root.

Value-oriented prioritization (VOP) technique follows the basic ordinal scale, and assesses requirements regarding their importance in the center production weight [24]. It also manages the weighting and identification of production reservations and the execution costs of each REQ. It builds a prioritization table by utilizing center production values,

REQs, and reservations. A few dimensions are not considered, such as the resources (or effort) required for every REQ. This research proposes a specific framework based on integrating the AHP and Grouping models. The AHP and Grouping Models are selected due to the attributes these models possess [25]. AHP is preferred as it is a consistent, proficient technique [26–28]. Numerical Grouping is the most used practice endorsed in IEEE Standard 830-1998 and RFC [29,30].

The primary objective of this research is to critically examine the existing methods of requirements prioritization to identify and address their deficiencies. Specifically, this study aims to enrich the existing body of knowledge for requirements prioritization by providing a detailed, comprehensive understanding of the fundamental terminologies and concepts prevalent in this domain. Our in-depth discussion covers a range of prioritization techniques and practices traditionally employed in software development, highlighting their respective dimensions and inherent challenges. A significant gap identified in the literature is the lack of a robust, quantitative approach in current prioritization methods, such as the commonly used grouping method, which merely categorizes requirements, without assigning quantifiable values. Moreover, while the Analytical Hierarchical Process (AHP) is a widely respected method for managing requirements, it struggles with scalability when dealing with large sets of requirements. These findings underscore the need for a more dynamic, scalable approach capable of handling complex, large-scale prioritization scenarios without compromising as to the accuracy or efficacy of the prioritization process. Our framework proposes solutions to these critical gaps, aiming to significantly enhance the precision and applicability of prioritization methods in contemporary software development environments, especially those involving complex systems like IoT applications.

3. Proposed Framework

A detailed description of the proposed framework is given in this section. This research proposes a specific framework based on an integration of the AHP and Grouping models. The proposed framework improves and enhances the existing approaches in order to overcome the limitations of existing models. The proposed framework comprises four modules: Pre-Processing, Review, Implementation, and Presentation Modules as depicted in Figure 1. The implementation module comprises a Grouping Processing Unit (GPU), which is used to classify the requirements into four groups, and an AHP Processing Unit (APU) responsible for prioritization. The AHP is favored because it is consistent with the reasonable requirements set. The proposed model deals with a sensible set of requirements, such as up to 48 requirements at one time. The APU will work on a FIFO basis if the requirement set is higher than 48. It is an iterative and incremental model.

In IoT-enabled applications, the requirements for presentation are always irregular. Tasks utilize a specific method from the various prioritization patterns, e.g., BST, AHP, Voting, etc. In the proposed model, the stakeholders' requirements will be negotiated before prioritization. In this study, we assume no mutual dependency among requirements. The assumption of no mutual dependency among requirements simplifies the analytical process within the prioritization framework, which is particularly useful in complex IoT systems where multiple interdependent factors could complicate decision-making. Isolation of requirements allows for a straightforward application of the Analytical Hierarchical Process (AHP) and advanced grouping methods, focusing on individual significance rather than interrelations. This assumption significantly reduces computational complexity, a critical advantage when integrating the extended AHP, in which the number of comparisons increases quadratically with the number of requirements. It also enhances the system's modularity, which is crucial in rapidly evolving IoT environments, by allowing modifications without a full system re-evaluation. The assumption of independence facilitates scalability in handling larger datasets and is practical in the early stages of technology development, aiding in establishing a solid framework foundation. The effectiveness of this approach is corroborated by case studies that show minimal negative impacts from ignoring dependencies, supporting its validity in practical application.

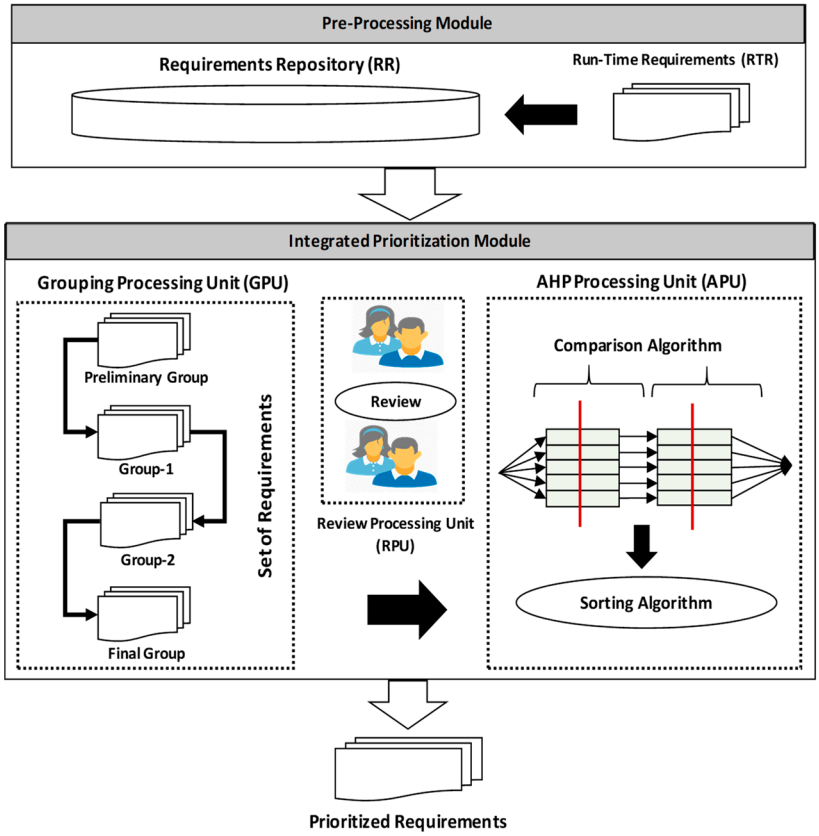

Figure 1. Proposed frameworks.

Two techniques are combined, i.e., AHP and Grouping (numerical assignment). No mutual prioritization exists within the Grouping, while AHP fails when given a vast set of requirements to match. The suggested combined model is developed to deal with the issues of both AHP and Grouping and to obtain the benefits of each procedure. The combined pattern has two phases (the pre-processing phase and the prioritization phase); these phases eliminate the pattern's difficulty and formulate the pattern in comprehensible terms. In every phase, one or more activities are accomplished, an element which will be discussed in the upcoming sections.

3.1. Pre-Processing Module

The pre-processing phase is the first phase of the suggested framework, in which the collected requirements are acknowledged. It comprises two parts, the requirements repository (RR) and run-time requirements (RTR), which are collected during development.

3.1.1. Requirements Repository (RR)

This is a database where requirements are stored and listed after elicitation. In this phase, the inappropriately elaborated requirements are discussed (procedure of negotiating ambiguous requirements with customers) by contacting (through forums, emails, wikis, etc.) customers (as customers are vital sources). The discussed requirements are then summed up to the RR.

3.1.2. Run-Time Requirements (RTR)

This is the second part of the preparation phase. It incorporates the incoming requirements during development. Requirements are continuously added to RR using a straightforward rule:

- The specific REQ is checked in the repository,
    - If a specific requirement already exists, its value is increased by 1.

- Else, add to the database.

This phase outcome is considered an input for the execution and reviewing phase.

3.2. Integrated Prioritization Module (IPM)

The Integrated Prioritization Module (IPM) is a pivotal component of our proposed framework, playing a crucial role in the overall prioritization process. This module is strategically designed to optimize requirement prioritization through a structured approach which is segmented into three distinct subunits: the Grouping Processing Unit (GPU), the Review Processing Unit (RPU), and the Analytical Hierarchical Process (AHP) Processing Unit (APU). Each of these subunits addresses specific aspects of the prioritization process:

3.2.1. Grouping Processing Unit (GPU)

Once the implementation module receives the requirements, they are moved to the GPU. The GPU groups these requirements based on resource availability, importance, and development. The classification is divided into four groups: Preliminary, Group 2, Group 3, and Final. Each category can store a maximum of 12 requirements, and the maximum number of requirements is limited within every category (such as 25% requirements). The 12×12 matrixes can be easily handled in AHP Engine. By allowing a small amount of time, more requirements are met. For instance, if we have fifty requirements instead of 50×50, there would be four 12×12 matrices. It is easy to perform cross-assessment of 12×12 matrices.

The limit of 12 requirements per group in our framework is strategically set to manage the complexity and computational efficiency within the AHP. AHP necessitates pairwise comparisons, within which the number of comparisons increases quadratically with the number of items. Specifically, for n items, the number of required pairwise comparisons is $n(n-1)/2$. By capping each group at 12, the pairwise comparisons are kept to a manageable 66 per group, which significantly enhances the computational feasibility of the AHP, ensuring that the prioritization remains both manageable and effective without overwhelming the system or the analysts. When the number of requirements exceeds 50, the FIFO approach is employed to ensure that the processing of requirements remains systematic and timely. This method is particularly useful in dynamic environments typical of IoT applications, where requirements can frequently accumulate. FIFO facilitates a fair and orderly processing sequence, ensuring that older requirements are addressed before newer ones, and thus preventing any requirement from being indefinitely postponed or ignored. However, these methods come with their potential risks and limitations. While the 12-requirements cap aids in simplifying the prioritization process, it might lead to oversimplification, particularly when there are complex interdependencies between requirements that might span across different groups. This could potentially dilute the precision of prioritization if significant interdependencies are overlooked. Moreover, while FIFO promotes fairness and systematic processing, it could also result in delays for requirements that enter the queue later, particularly if there is a continuous influx of new requirements. This might slow down the response to urgent or evolving needs if not adequately monitored and managed.

The requirements from RR are moved to the categories (Preliminary Group, Group 2, Group 3, and Final Group) in the following manner:

- The first 12 highly important REQs are sent to the Final Group;
- Subsequently, 12 important REQs are shifted to Group 3;
- The next 12 major REQs are sent to Group 2;
- The last 12 significant REQs have been moved to the Preliminary Group.

This means that the GPU initially performs grouping without calculation. The initial grouping is performed based on expert opinion, and the proposed system has nothing to do with the initial grouping level.

The movement of requirements between groups in our dynamic prioritization process is a carefully considered strategy designed to adapt to the evolving criticalities and depen-

dencies of requirements identified during the review phase. This flexibility is essential in IoT environments, in which project scopes and stakeholder needs can change rapidly. We acknowledge that such movements could potentially disrupt the balance of groups in terms of their requirements capacities. To ensure that the prioritization accurately reflects the most current evaluations of each requirement's importance and urgency, movements between groups are allowed in order to align the groups with the latest insights.

To manage the balance of groups after such movements, we have implemented a specific balancing mechanism within our algorithm. Following any movement of requirements, we conduct an assessment to ensure that no group's capacity is exceeded by more than 10%. Should this threshold be breached, a rebalancing is triggered, which may involve adjusting the criteria that define each group's boundaries or transferring some requirements to less-dense groups.

3.2.2. Reviewing Processing Unit (RPU)

Following the initial grouping of requirements, a thorough review is essential, especially since the categorization is often performed by individuals who may not have technical expertise, such as users of IoT-enabled smart city services. This review process involves domain experts, developers, and testers—all of whom have a deep understanding of the application designs relevant to the requirements. During this phase, these professionals work collaboratively to adjust and realign the requirement groupings. Any issues related to inappropriate grouping are addressed; for example, if a requirement like R-1 is incorrectly placed in the last group—which is reserved for critical requirements—but actually belongs in Group 2, it will be corrected by moving the requirement to the appropriate group. This ensures that all requirements are accurately categorized for effective prioritization.

3.2.3. AHP Processing Unit (APU)

The APU is preset in the GPU Final Group. This unit is purely based on AHP. AHP is preferred because it is a reliable technique that meets many requirements. The number of requirements covered by our proposed model is up to 50. The APU will work on a FIFO basis if the requirement exceeds 100. It will process and manage only 48 requirements at one time. It is an iterative and incremental model. The simple aim of the APU is to allocate a proper priority to every specific REQ listed in the groups. This will overcome the misperceptions associated with the question of which REQ to execute first. The outcome of the APU will be the final outcome, including the list of prioritized REQs. The result of the APU will be required to be accurate and have actual priority so it can be executed to develop and design the application without any misunderstandings. The algorithm of the proposed framework is as follows:

1. The initial requirements are stored in a repository.
    - This should be explored as three keywords: for instance, Requirement/Need/Feature;
    - This should be distinct in three words: Object/Action/Result.
2. A specific requirement is not entered again, but its value/importance is incremented by one if the requirement already exists in the repository.
    - It is just added to the repository if it is not already present in the repository.
3. The requirements listed in the repository are argued based on the win–win pattern.
4. Additional requirements coming at run-time during the design process are added to the repository using step 2.
5. Primarily, 40–48 maximum requirements (12 requirements in the four different groups) should be adequately defined and then sent to the prioritization process from the repository. These requirements are initially moved to the GPU. The 12-requirement set is demonstrated in a 12×12 table (matrix) in such a form that it denotes the distinctive pairs. This matrix represents 12 requirements per group. Four groups will entertain 48 requirements at the same time.

6. Requirements are manually prioritized in the GPU based on the significance of the practicality of execution/designing it and the availability of resources.
    - The manual process is used because the elicitation of requirements is a manual method.
7. The requirements are categorized into four groups: Preliminary, Group 2, Group 3, and Final.
8. Afterwards, the domain expert (such as a programmer/developer) inspects all the requirements regarding feasibility and importance.
    - The problems concerning grouping and categorization are resolved in this step;
    - The misplaced requirements are moved to their correct place.
9. The APU is fixed in the most serious group, which scans serious and important requirements and sorts only the important requirements, with respect to the importance, cost, and risk associated with every requirement in this group.
    - The prioritized (sorted) requirements are then moved for development and execution.
10. The requirements from the Preliminary Group are extracted to Group 2, Group 2 to Group 3, and Group 3 to the Final Group at particular intervals to avoid severe issues, such as the danger of starvation.
11. If two or more than 2 REQs have equal priorities once the APU prioritization process is complete
    - The REQs are moved for execution and development using the FIFO approach.

4. Validation

The justification and validation of the proposed model, together with a detailed description, is presented in this section. Specifically, it explains the most critical components defined by the proposed framework. Case studies test the validity and applicability of the proposed framework. Furthermore, the proposed framework is developed using a Design and Creation approach, and thoroughly designed to bridge the gaps identified in existing requirement prioritization methods for IoT applications. It is informed by a thorough review of the relevant literature and existing work, ensuring that our framework integrates and builds upon established best practices and recent advancements in the field. The strategy of Design and Creation emphasizes the design of new products [30]. The proposed framework is composed of constructs and models. Construct is the vocabulary used in a particular domain, and model is the conceptual representation supporting problem comprehension and solving.

The comparison scales listed in Table 1 are utilized to allocate significance at every matrix cell. In the literature, this scale is used by [17]. This table compares the importance of the requirements. If two requirements are equally important, then weight one will be selected. If the difference in importance is moderate between the two requirements, then weight three will be selected. Similarly, the required weight is selected for essential, major, and extreme differences. This scale establishes which REQ is more important than the other. In addition, it also illustrates the degree to which one REQ is more significant than another. Values (1–9) are allocated across every cell, revealing the variation in the significance of 1 REQ relative to others. Let us suppose we have a set of REQs, such as REQ1, REQ2, . . . REQ24. Considering the user's importance, these REQs should be prioritized, such as Preliminary Group, Group 2, Group 3, and Final Group, as shown in Table 2A.

Independent developers from various external software companies, collectively referred to for illustrative purposes as 'Software House ABC', are involved in inspecting and assessing the proposed framework. These developers provide critical insights and feedback, ensuring that the framework is robust and applicable across different technological environments and use cases. The developers inspect the REQs, and the groups are subject to variation due to some of the REQs that were not appropriately positioned (such as REQ7 and REQ8 being moved from Group 3 to the Final Group; REQ13 and REQ14 being moved to Group 3 from Group 2; and REQ19 and REQ20 being moved from the Preliminary Group

to Group 2). Afterward, the REQs are placed correctly. The updated groups are depicted in Table 2B. REQs in the Final Group are prioritized (supposition) using the Table 1 scale, as shown in Table 3A.

Table 1. Comparison scale (pairwise) [17].

Description	Weight (W)
Equal Importance	01
Intermediate value between 1 and 3	02
Moderate difference in importance	03
Intermediate value between 3 and 5	04
An essential difference in importance	05
Intermediate value between 5 and 7	06
A significant difference in importance	07
Intermediate value between 7 and 9	08
An extreme difference in importance	09

Table 2. (**A**): Grouping using proposed framework before review. (**B**): Grouping using proposed framework after review.

(**A**)		
S#	**Group**	**REQs**
01	Final Group	REQ1, REQ2, REQ3, REQ4, REQ5, REQ6
02	Group 3	REQ7, REQ8, REQ9, REQ10, REQ11, REQ12
03	Group 2	REQ13, REQ14, REQ15, REQ16, REQ17, REQ18
04	Preliminary Group	REQ19, REQ20, REQ21, REQ22, REQ23, REQ24
(**B**)		
S#	**Group**	**REQs**
01	Final Group	REQ1, REQ2, REQ3, REQ4, REQ5, REQ6, REQ7, REQ8
02	Group 3	REQ9, REQ10, REQ11, REQ12, REQ13, REQ14
03	Group 2	REQ15, REQ16, REQ17, REQ18, REQ19, REQ20
04	Preliminary Group	REQ21, REQ22, REQ23, REQ24

The subsequent activity comprises normalizing the "rows_sum" column and determining a percentage from the value of the total column. The significance figure of REQi*j is divided by the value for "columns sum". The sum of every row is calculated for the next phase [17]. Table 3B demonstrates the rows_sum.

Afterward, the row_sum of every row is divided by the total number of REQs, which is 8. Hence, we obtained each REQ's priority, as shown in Table 3C (e.g., Eigenvalues) [17].

Lastly, the REQs and their respective priorities are transmitted to the implementation module. REQs having equal priorities are entertained using the FIFO approach. After managing the Final Group, the requirements repository (RR) is looked to for new REQs. In

this scenario, REQ25, REQ26, REQ27, REQ28, REQ29. . .. REQ30 are the REQs that are most significant, as declared by the users at run-time. Therefore, these REQs will be transferred to the Grouping Processing Unit (GPU) Final Group and reviewed, in order to eliminate inconsistency. Rather than grouping only three REQs, these late-declared REQs are sent to the AHP Processing Unit (APU), and run-time REQs are managed this way. The tabulated form is shown in Table 4A.

Table 3. (**A**): Prioritization of 8 REQs via APU within the proposed framework. (**B**): Horizontal sum of values from the array. (**C**): Priorities assigned via APU within the proposed framework.

(**A**)

REQ#	REQ1	REQ2	REQ3	REQ4	REQ5	REQ6	REQ7	REQ8
EQ1	1.00	5.00	3.00	3.00	9.00	7.00	9.00	9.00
REQ2	0.20	1.00	5.00	5.00	7.00	7.00	7.00	3.00
REQ3	0.33	0.20	1.00	3.00	9.00	7.00	9.00	9.00
REQ4	0.33	0.20	0.33	1.00	9.00	7.00	9.00	9.00
REQ5	0.11	0.14	0.11	0.11	1.00	3.00	3.00	9.00
REQ6	0.14	0.14	0.14	0.14	0.33	1.00	3.00	9.00
REQ7	0.11	0.14	0.11	0.11	0.33	0.33	1.00	9.00
REQ8	0.11	0.33	0.11	0.11	0.11	0.11	0.11	1.00
Total	2.34	7.16	9.81	12.48	35.78	32.44	41.11	58.00

(**B**)

REQ#	REQ1	REQ2	REQ3	REQ4	REQ5	REQ6	REQ7	REQ8	SUM
REQ1	0.43	0.70	0.31	0.24	0.25	0.22	0.22	0.16	2.51
REQ2	0.09	0.14	0.51	0.40	0.20	0.22	0.17	0.05	1.77
REQ3	0.14	0.03	0.10	0.24	0.25	0.22	0.22	0.16	1.35
REQ4	0.14	0.03	0.03	0.08	0.25	0.22	0.22	0.16	1.13
REQ5	0.05	0.02	0.01	0.01	0.03	0.09	0.07	0.16	0.44
REQ6	0.06	0.02	0.01	0.01	0.01	0.03	0.07	0.16	0.38
REQ7	0.05	0.02	0.01	0.01	0.01	0.01	0.02	0.16	0.29
REQ8	0.05	0.05	0.01	0.01	0.00	0.00	0.00	0.02	0.14

(**C**)

REQ#	REQ1	REQ2	REQ3	REQ4	REQ5	REQ6	REQ7	REQ8	Priorities
REQ1	0.43	0.70	0.31	0.24	0.25	0.22	0.22	0.16	0.31
REQ2	0.09	0.14	0.51	0.40	0.20	0.22	0.17	0.05	0.22
REQ3	0.14	0.03	0.10	0.24	0.25	0.22	0.22	0.16	0.17
REQ4	0.14	0.03	0.03	0.08	0.25	0.22	0.22	0.16	0.14
REQ5	0.05	0.02	0.01	0.01	0.03	0.09	0.07	0.16	0.05
REQ6	0.06	0.02	0.01	0.01	0.01	0.03	0.07	0.16	0.05
REQ7	0.05	0.02	0.01	0.01	0.01	0.01	0.02	0.16	0.04
REQ8	0.05	0.05	0.01	0.01	0.00	0.00	0.00	0.02	0.02
									1.00

Table 4. (**A**): Additional processing of REQs via APU within the proposed framework. (**B**): Horizontal sum of values from the array. (**C**): Priorities assigned via APU within the proposed framework.

(**A**)						
REQ#	**REQ25**	**REQ26**	**REQ27**	**REQ28**	**REQ29**	**REQ30**
REQ25	1.00	5.00	3.00	5.00	5.00	5.00
REQ26	0.20	1.00	5.00	3.00	3.00	3.00
REQ27	0.33	0.20	1.00	5.00	5.00	5.00
REQ28	0.20	0.33	0.20	1.00	3.00	3.00
REQ29	0.20	0.33	0.20	0.33	1.00	3.00
REQ6	0.20	0.33	0.20	0.33	0.33	1.00
SUM	2.13	7.20	9.60	14.67	17.33	20.00

(**B**)							
REQ#	**REQ25**	**REQ26**	**REQ27**	**REQ28**	**REQ29**	**REQ30**	**SUM**
REQ25	0.47	0.69	0.31	0.34	0.29	0.25	2.36
REQ26	0.09	0.14	0.52	0.20	0.17	0.15	1.28
REQ27	0.16	0.03	0.10	0.34	0.29	0.25	1.17
REQ28	0.09	0.05	0.02	0.07	0.17	0.15	0.55
REQ29	0.09	0.05	0.02	0.02	0.06	0.15	0.39
REQ6	0.09	0.05	0.02	0.02	0.02	0.05	0.25

(**C**)							
REQ#	**REQ25**	**REQ26**	**REQ27**	**REQ28**	**REQ29**	**REQ30**	**P**
REQ25	0.47	0.69	0.31	0.34	0.29	0.25	0.39
REQ26	0.09	0.14	0.52	0.20	0.17	0.15	0.21
REQ27	0.16	0.03	0.10	0.34	0.29	0.25	0.19
REQ28	0.09	0.05	0.02	0.07	0.17	0.15	0.09
REQ29	0.09	0.05	0.02	0.02	0.06	0.15	0.07
REQ6	0.09	0.05	0.02	0.02	0.02	0.05	0.04
							1.00

The subsequent activity comprises normalizing the "rows_sum" column, which determines a percentage from the column's total value. The significance of the figure for REQi*j is divided by the "columns sum" value depicted in Table 4B.

Subsequently, the priorities are determined by dividing each value in the "rows _sum" column by the total number of requirements (REQs) in the group. This calculation assigns a priority level to each requirement, with the results displayed in Table 4C. In this table, the "priority" column indicates the relative importance of each specific REQ. After these priorities are established, the REQs are forwarded to the next module for implementation.

The method then rechecks RR for new important REQs where there are no new REQs in the current scenario. Therefore, the REQs in Group 3 are transferred to the APU for comparison. Group 3 includes REQ9, REQ10, REQ11, REQ12, REQ13, and REQ14, as shown in Table 5A.

The calculation of percentage values is revisited by dividing each value by the sum of its respective column, followed by computing the sum for each row, as depicted in Table 5B. Subsequently, each value in the "rows_sum" column is divided by the total number of requirements (REQs), which is 6, to determine the priority of each REQ, as shown in Table 5C. At this point, the REQs are forwarded to the implementation stage, where they are processed based on their assigned priorities.

Table 5. (**A**): Group 3 prioritization via APU within the proposed framework. (**B**): Horizontal sum of values from the array. (**C**): Priorities assigned via APU within the proposed framework.

(**A**)						
REQ#	**REQ9**	**REQ10**	**REQ11**	**REQ12**	**REQ14**	**REQ14**
REQ9	1.00	5.00	3.00	5.00	5.00	3.00
REQ10	0.20	1.00	3.00	5.00	5.00	3.00
REQ11	0.33	0.33	1.00	5.00	5.00	3.00
REQ12	0.20	0.20	0.20	1.00	5.00	3.00
REQ14	0.20	0.20	0.20	0.20	1.00	3.00
REQ14	0.33	0.33	0.33	0.33	0.33	1.00
SUM	2.27	7.07	7.73	16.53	21.33	16.00

(**B**)							
REQ#	**REQ9**	**REQ10**	**REQ11**	**REQ12**	**REQ14**	**REQ14**	**SUM**
REQ9	0.44	0.71	0.39	0.30	0.23	0.19	2.26
REQ10	0.09	0.14	0.39	0.30	0.23	0.19	1.34
REQ11	0.15	0.05	0.13	0.30	0.23	0.19	1.05
REQ12	0.09	0.03	0.03	0.06	0.23	0.19	0.62
REQ14	0.09	0.03	0.03	0.01	0.05	0.19	0.39
REQ14	0.15	0.05	0.04	0.02	0.02	0.06	0.34

(**C**)							
REQ#	**REQ9**	**REQ10**	**REQ11**	**REQ12**	**REQ14**	**REQ14**	**P**
REQ9	0.44	0.71	0.39	0.30	0.23	0.19	0.38
REQ10	0.09	0.14	0.39	0.30	0.23	0.19	0.22
REQ11	0.15	0.05	0.13	0.30	0.23	0.19	0.17
REQ12	0.09	0.03	0.03	0.06	0.23	0.19	0.10
REQ14	0.09	0.03	0.03	0.01	0.05	0.19	0.06
REQ14	0.15	0.05	0.04	0.02	0.02	0.06	0.06
							1.00

Concurrently, a review of the requirement repository (RR) is conducted to ascertain if any REQs are listed in the Final Group or Group 3, which would necessitate their transfer to the Grouping Processing Unit (GPU) for further action. In the current scenario, no REQs are found in the Final Group or Group 3 within the RR. Therefore, REQs from Group 2 are directed to the Allocation Processing Unit (APU) for evaluation. Group 2 comprises REQ15, REQ16, REQ17, REQ18, REQ19, and REQ20, as shown in Table 6A.

The following task involves normalizing the sum of each row and calculating a percentage based on the total value of the column. Each cell's substantial cost is then distributed proportionally across the column's sum, as illustrated in Table 6B.

Afterward, every value in the column of "rows_sum" is divided into the total number of requirements (which is 6) in order to obtain priorities for every REQ shown in Table 6C.

These REQs are, at this step, sent to the implementation stage, where every REQ is processed according to priority. At this point, the RR (requirement repository) is checked again to determine whether REQs are presented in the Final Group, Group 3, or Group 2; they would be then transferred to the Grouping Processing Unit (GPU), and the same activity would be carried out. In our current scenario, no REQ is present in the Final Group, Group 3, or Group 2 of RR. Therefore, REQs from the Preliminary Group are sent to the

APU for comparison. The Preliminary Group includes REQ21, REQ22, REQ23, and REQ24, as shown in Table 7A.

Table 6. (**A**): Group 2 prioritization via APU within the proposed framework. (**B**): Horizontal sum of values from the array. (**C**): Priorities assigned via APU within the proposed framework.

(**A**)						
REQ#	**REQ15**	**REQ16**	**REQ17**	**REQ18**	**REQ19**	**REQ20**
REQ15	1.00	3.00	3.00	5.00	3.00	5.00
REQ16	0.33	1.00	3.00	5.00	3.00	3.00
REQ17	0.33	0.33	1.00	5.00	3.00	3.00
REQ18	0.20	0.20	0.20	1.00	3.00	3.00
REQ19	0.33	0.33	0.33	0.33	1.00	5.00
REQ20	0.20	0.33	0.33	0.33	0.20	1.00
SUM	2.40	5.20	7.87	16.67	13.20	20.00

(**B**)							
REQ#	**REQ15**	**REQ16**	**REQ17**	**REQ18**	**REQ19**	**REQ20**	**SUM**
REQ15	0.42	0.58	0.38	0.30	0.23	0.25	2.15
REQ16	0.14	0.19	0.38	0.30	0.23	0.15	1.39
REQ17	0.14	0.06	0.13	0.30	0.23	0.15	1.01
REQ18	0.08	0.04	0.03	0.06	0.23	0.15	0.58
REQ19	0.14	0.06	0.04	0.02	0.08	0.25	0.59
REQ20	0.08	0.06	0.04	0.02	0.02	0.05	0.27

(**C**)							
REQ#	**REQ15**	**REQ16**	**REQ17**	**REQ18**	**REQ19**	**REQ20**	**P**
REQ15	0.42	0.58	0.38	0.30	0.23	0.25	0.36
REQ16	0.14	0.19	0.38	0.30	0.23	0.15	0.23
REQ17	0.14	0.06	0.13	0.30	0.23	0.15	0.17
REQ18	0.08	0.04	0.03	0.06	0.23	0.15	0.10
REQ19	0.14	0.06	0.04	0.02	0.08	0.25	0.10
REQ20	0.08	0.06	0.04	0.02	0.02	0.05	0.05
							1.00

Table 7. (**A**): Preliminary Group prioritization via APU within the proposed framework. (**B**): Horizontal sum of values from the array. (**C**): Priorities assigned via APU within the proposed framework.

(**A**)				
REQ#	**REQ21**	**REQ22**	**REQ23**	**REQ24**
REQ21	1.00	5.00	3.00	3.00
REQ22	0.20	1.00	3.00	5.00
REQ23	0.33	0.33	1.00	3.00
REQ24	0.33	0.20	0.33	1.00
SUM	1.87	6.53	7.33	12.00

Table 7. *Cont.*

(**B**)					
REQ#	**REQ21**	**REQ22**	**REQ23**	**REQ24**	**SUM**
REQ21	0.53	0.77	0.41	0.25	1.96
REQ22	0.11	0.15	0.41	0.42	1.09
REQ23	0.18	0.05	0.14	0.25	0.62
REQ24	0.18	0.03	0.05	0.08	0.34
(**C**)					
REQ#	**REQ21**	**REQ22**	**REQ23**	**REQ24**	**P**
REQ21	0.53	0.77	0.41	0.25	0.49
REQ22	0.11	0.15	0.41	0.42	0.27
REQ23	0.18	0.05	0.14	0.25	0.15
REQ24	0.18	0.03	0.05	0.08	0.08
SUM					1.00

The next step involves normalizing the sum of each row and calculating a percentage relative to the total of each column. Specifically, the value of each cell is divided by the sum of its corresponding column, as demonstrated in Table 7B.

Subsequently, every value in the "rows_sum" column is divided by the total number of REQs (which is 4) to obtain the priorities for every REQ, as shown in Table 7C.

Table 8 provides a comparative analysis with the proposed framework and other existing techniques. The proposed framework deals with Objective Grouping, Pair-Wise Comparison, Ratio Scale, Hierarchy Level, Numerical Assignment, Ordinal Scale, Four Groups, Stakeholder Management, and Same Priority Management, while the existing techniques do not provide these management attributes.

Table 8. Comparative analysis.

Existing Techniques	**Proposed Framework**
Subjective Grouping	Objective Grouping
No Pair-Wise Comparison	Pair-Wise Comparison
No Ratio Scale	Ratio Scale
Hierarchy Level Management	Hierarchy Level
Subjective Assignment	Numerical Assignment
No Ordinal Scale	Ordinal Scale
Three Groups	Four Groups
Lack of stakeholder Management	Stakeholder Management
Not Dealing with the Same Priority	Same-Priority Management

4.1. Discussion

The proposed framework, integrating the Extended Analytical Hierarchical Process (AHP) with an advanced grouping model, was rigorously tested through a series of case studies specifically designed to evaluate its effectiveness in IoT application development. These case studies were carefully selected based on their diversity in scope and complexity, as well as the variety of IoT technologies they employed, ensuring a comprehensive assessment of the framework across different scenarios. Each case study involved a detailed analysis of the requirements prioritization process for an IoT application, from initial collection through to final prioritization. The studies were conducted in simulated environments

that replicated real-world IoT systems, encompassing varying degrees of requirement complexity and interdependencies. The primary criteria for evaluation included the efficiency of the prioritization process, the accuracy of requirement classification, and the overall impact on project outcomes.

4.2. Findings and Observations

- Efficiency Improvements: The integration of an enhanced AHP with a systematic grouping approach significantly reduced the time required for requirements analysis and prioritization. On average, the time savings amounted to approximately 30%, compared to traditional methods, as demonstrated across multiple case studies.
- Accuracy in Requirement Classification: The advanced grouping model proved particularly effective in categorizing requirements into groups that were more nuanced than the conventional tripartite division (inessential, desirable, and mandatory). This nuanced categorization allowed for a more precise assessment of each requirement's impact, leading to more informed decision-making processes.
- Enhanced Design Quality: The application of the framework led to observable improvements in design quality and system functionality. By accurately prioritizing critical requirements, development teams were able to allocate resources more effectively, enhancing the robustness and performance of the final IoT solutions.

The complexity of integrating AHP within large-scale IoT projects remains a challenge, particularly when dealing with highly interconnected and dynamic environments. Additionally, the dependency on expert input for initial requirement grouping may introduce a degree of subjectivity, which could affect the replicability of the process across different teams or projects.

4.3. Comparative Analysis

Table 8 in the paper presents a comparative analysis between existing prioritization techniques and the newly proposed framework. The existing methods are characterized by subjective grouping, while requirements are typically categorized without a standardized criterion, leading to potential inconsistencies. In contrast, the proposed framework utilizes objective grouping, which bases the categorization on defined, measurable criteria, enhancing consistency and transparency. Existing techniques often lack a structured pairwise comparison, while this is a methodological feature of the proposed framework. This enables a more detailed and comparative analysis of requirements, helping to establish clearer priorities. Similarly, while traditional methods do not employ a ratio scale, the proposed framework integrates this scale to quantify the differences in importance between requirements more accurately. The management of hierarchy levels, which is absent or unstructured in current practices, is systematically addressed in the proposed framework. This structured approach allows for clearer definition and understanding of requirement layers and their interdependencies.

Furthermore, while existing methods use subjective assignment of requirements to categories, the proposed framework adopts numerical assignment, which enhances the precision of requirement categorization. Traditional techniques often do not use an ordinal scale, while the proposed framework incorporates this aspect, allowing for the ranking of requirements in a standardized manner. This addition helps in systematically determining the relative importance of each requirement. Additionally, existing methods typically organize requirements into three broad groups. The proposed framework expands this by introducing a fourth group, enabling a more nuanced categorization which better addresses the complexity of IoT systems. The proposed framework also addresses the lack of stakeholder management in current methods by explicitly incorporating stakeholder inputs into the requirement prioritization process, ensuring that all stakeholder needs are considered. Lastly, the proposed framework includes mechanisms for managing requirements with the same priority, an aspect often overlooked in existing methods. This feature ensures that

critical requirements are not overlooked simply because they share similar priority levels, thereby supporting more effective project management and execution.

5. Challenges and Limitations

To explore the challenges and limitations associated with implementing the proposed framework for optimizing requirements prioritization in IoT applications, it is essential to consider both practical and theoretical aspects.

5.1. Practical Challenges

- Integration Complexity: Integrating the proposed framework with existing systems can be challenging. It requires significant customization to accommodate the specific needs of different IoT applications, which may vary widely in their complexities and operational contexts.
- Scalability Concerns: While the framework aims to handle a broad range of requirements efficiently, scaling it for extremely large datasets or highly complex projects may require additional optimization to prevent performance bottlenecks, especially in the Analytical Hierarchical Process (AHP), when dealing with many pairwise comparisons.
- Stakeholder Alignment: Gaining consensus among stakeholders can be a significant hurdle, particularly in environments with diverse interests and priorities. The framework's reliance on stakeholder input for prioritizing requirements can lead to delays if not managed efficiently.

5.2. Theoretical Limitations

- Dependency Management: The framework assumes a lack of mutual dependency among requirements in some of its phases, which might not always be the case in complex IoT systems, where requirements are often interdependent. This simplification could lead to suboptimal prioritization outcomes if dependencies are critical.
- Subjectivity in Grouping: Even with an advanced grouping methodology, the subjective nature of categorizing requirements into essential, desirable, or inessential might still persist to some degree. This subjectivity can influence the prioritization process, especially in the initial classification stage.
- Adaptability to Changes: IoT systems are dynamic, with requirements that can evolve rapidly due to technological advancements or changing market conditions. The framework's ability to adapt to these changes efficiently without extensive reconfiguration remains a critical area for further development.

5.3. Potential for Future Enhancements

- Automated Dependency Recognition: Enhancing the framework by adding the capability to automatically detect and adapt to requirement dependencies using machine learning algorithms could significantly improve its effectiveness.
- Real-Time Data Integration: Incorporating real-time data analytics to continuously update and prioritize requirements based on ongoing feedback and system performance metrics would be of value.
- Advanced Simulation Tools: Developing simulation tools that can model the impact of different prioritization strategies on the overall system performance could help in fine-tuning the framework before full-scale implementation.

Addressing these concerns through continuous research and development will be crucial in enhancing its applicability and effectiveness in real-world scenarios. Further studies could focus on integrating adaptive algorithms and real-time data processing to ensure the framework remains robust and flexible in the face of rapidly changing technology landscapes.

6. Conclusions

This study presents a novel integrated prioritization framework specifically designed to address the unique challenges of requirement prioritization within IoT applications.

By analyzing the limitations inherent in existing prioritization methods, both in closed-source and open-source environments, this research contributes a robust solution that enhances adaptability and efficiency in managing IoT requirements. Unlike traditional methods which often struggle with handling equivalent priority requirements and adapting to changes in requirements during development, the proposed framework introduces dual-scale prioritization (ratio and ordinal) and facilitates prototyping post-single group prioritization—features not typically found in conventional models like AHP. Additionally, the framework's design allows for dynamic adjustments to the prioritization process in response to real-time changes, supporting continuous integration of new requirements without the need to restart the process. It also effectively manages larger sets of requirements by segmenting them into manageable groups, thereby overcoming the complexities associated with large NxN matrices. Through these innovations, the framework not only addresses current gaps but also sets the stage for future enhancements in the field of IoT requirement engineering.

Author Contributions: Conceptualization, S.K., M.A. and M.B.; Methodology, S.K., M.A. and M.B.; Software, S.K., M.A. and M.B.; Validation, M.E.-A. and M.B.; Formal analysis, S.K., M.A. and M.B.; Investigation, M.A. and M.B.; Resources, M.E.-A. and M.B.; Writing—original draft, S.K.; Writing—review and editing, M.A., M.E.-A. and M.B.; Supervision, M.E.-A. and M.B.; Funding acquisition, M.A. and M.E.-A. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to thank Prince Sultan University for paying the APC of this article.

Data Availability Statement: All the data is available in the article. The data presented in this study are available on request from the corresponding author.

Acknowledgments: This work was supported by EIAS Data Science Lab, CCIS, Prince Sultan University. The Grammarly tool was used to prepare this document. It was used solely to check the grammar and enhance the English-language quality of the manuscript. Grammarly was used after the manuscript was completed to ensure the clarity, correctness, and conciseness of the language. Still, it had no role in the content's ideation or creation.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Nassereddine, M.; Khang, A. Applications of Internet of Things (IoT) in smart cities. In *Advanced IoT Technologies and Applications in the Industry 4.0 Digital Economy*; CRC Press: Boca Raton, FL, USA, 2024; pp. 109–136.
2. Badshah, A.; Ghani, R.; Haleem, F.; Anwar, G.; Shahid, S.; Muhammad, Z.; Moustafa, M.N. Transforming educational institutions: Harnessing the power of internet of things, cloud, and fog computing. *Future Internet* **2023**, *15*, 367. [CrossRef]
3. Alhenawi, E.; Awawdeh, S.; Khurma, R.A.; García-Arenas, M.; Castillo, P.A.; Hudaib, A. Choosing a Suitable Requirement Prioritization Method: A Survey. *arXiv* **2024**, arXiv:2402.13149.
4. Malik, A.; Nordin, A.; Al-Ehaidib, R. Requirements Engineering (RE) Process for the Adaptation of the Hospital Information System (HIS). Available online: https://core.ac.uk/download/pdfr/325990671.pdf (accessed on 28 April 2024).
5. Yaseen, M.; Ibrahim, N.; Mustapha, A. Requirements Prioritization and using Iteration Model for Successful Implementation of Requirements. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 121–127. [CrossRef]
6. Lencastre, M.; Silva, D.; Pimentel, J.H.C.; Castro, J.B. PRIUS: Applying Gamification to User Stories Prioritization. *ACM SIGAPP Appl. Comput. Rev.* **2024**, *23*, 27–44. [CrossRef]
7. Hujainah, F.; Abu Bakar, R.B.; Abdulgabber, M.A. Investigation of requirements interdependencies in existing requirements prioritization techniques. *Teh. Vjesn.* **2019**, *26*, 1186–1190.
8. Anwar, R.; Bashir, M.B. A Systematic Literature Review of AI-based Software Requirements Prioritization Techniques. *IEEE Access* **2023**, *11*, 143815–143860. [CrossRef]
9. Mohamed, A.S.I.; El-Maddah, B.I.A.; Wahba, C.A.M. Criteria based requirements prioritization for software products. In Proceedings of the 2008 International Conference on Software Engineering Research and Practice SERP, Las Vegas, NV, USA, 14–17 July 2008; pp. 587–593.
10. Rehman, S.U.; Aoun, M.; Qayoom, A. Selection Criteria for Requirement Prioritization Techniques for software development: Data Analysis. *Asian Bull. Big Data Manag.* **2023**, *3*, 201–215. [CrossRef]
11. Sher, F.; Jawawi, D.N.; Mohammad, R. Requirements Prioritization Aspects Quantification For Value-Based Software Developments. *J. Theor. Appl. Inf. Technol.* **2019**, *97*, 3969–3979.

12. Kiran, H.M.; Ali, Z. Requirement Elicitation Techniques for Open Source Systems: A Review. *Int. J. Adv. Comput. Sci. Appl. Pak.* **2018**, *9*, 330–334.
13. Bhowmik, T.; Do, A.Q. Refinement and resolution of just-in-time requirements in open source software and a closer look into non-functional requirements. *J. Ind. Inf. Integr.* **2019**, *14*, 24–33. [CrossRef]
14. Kuštelega, M.; Mekovec, R. A Systematic Analysis of Requirements Elicitation Problems and Challenges. In Proceedings of the Central European Conference on Information and Intelligent Systems, Varazdin, Croatia, 20–22 September 2023; pp. 465–471.
15. Stamelos, I.; Gonzalez-Barahoña, J.M.; Varlamis, I.; Anagnostopoulos, D. (Eds.) Open Source Systems: Enterprise Software and Solutions. In Proceedings of the 14th IFIP WG 2.13 International Conference, OSS 2018, Athens, Greece, 8–10 June 2018; Proceedings; Springer: Berlin/Heidelberg, Germany, 2018; Volume 525.
16. Noviyanto, F.; Razali, R.; Nazri, M.Z.A. Understanding requirements dependency in requirements prioritization: A systematic literature review. *Int. J. Adv. Intell. Inform.* **2023**, *9*, 249. [CrossRef]
17. Mao, G. Research on Customer Requirement Prioritization of Engineering Products Based on Group Multi-Granularity Linguistic Information. *Sci. Soc. Res.* **2024**, *6*, 85–90. [CrossRef]
18. Ayala, C.; Nguyen-Duc, A.; Franch, X.; Höst, M.; Conradi, R.; Cruzes, D.; Babar, M.A. System requirements-OSS components: Matching and mismatch resolution practices–an empirical study. *Empir. Softw. Eng.* **2018**, *23*, 3073–3128. [CrossRef]
19. Khan, K.A. A Systematic Literature Review of Software Requirements Prioritization. Master's Thesis, Blekinge Institute of Technology (BTH), Karlskrona, Sweden, 23 October 2006; pp. 41–43.
20. Babar, M.I.; Ramazan, M.; Ghayyur, S.A.K. Challenges and future trends in software requirements prioritization. In Proceedings of the International Conference on Computer Networks and Information Technology ICCNIT, Abbottabad, Pakistan, 11–13 July 2011; pp. 319–324.
21. Iqbal, M.A.; Zaidi, A.M.; Murtaza, S. A new requirements prioritization model for market driven products using AHP. In Proceedings of the International Conference on Data Storage and Date Engineering DSDE, Bangalore, India, 9–10 February 2010; pp. 142–149.
22. Tufail, H.; Qasim, I.; Masood, M.F.; Tanvir, S.; Butt, W.H. Towards the selection of Optimum Requirements Prioritization Technique: A Comparative Analysis. In Proceedings of the 2019 5th International Conference on Information Management (ICIM), Cambridge, UK, 24–27 March 2019.
23. Aasem, M.; Ramazan, M.; Jaffar, M. Analysis and optimization of software requirements prioritization techniques. In Proceedings of the International Conference on Information and Emerging Technologies ICIET, Karachi, Pakistan, 14–16 June 2010; pp. 1–6.
24. Iqbal, A.; Khan, F.M.; Khan, S.A. A critical analysis of techniques for requirement prioritization and open research issues. *Int. J. Rev. Comput. IJRIC* **2009**, *1*, 8–18.
25. Anna, P.; Filippo, R.; Angelo, S.; Cinzia, B. An Empirical Study to Compare the Accuracy of AHP and CBRanking Techniques for Requirements Prioritization. In Proceedings of the Fifth International Workshops on Comparative Evaluation in Requirements Engineering (CERE'07), New Delhi, India, 16 October 2007.
26. Carlos, E.O.; Luis, D.O. A Quality-Based Requirement Prioritization Framework Using Binary Inputs. In Proceedings of the Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, Kota Kinabalu, Malaysia, 26–28 May 2010.
27. Chuan, D.; Paula, L.; Cleland-Huang, J.; Kwiatkowski, C. To wards automated requirements prioritization triage. *Requir. Eng.* **2009**, *14*, 73–89.
28. Sorooshian, S.; Azizan, N.A.B. Expedited Analytical Hierarchical Process for Multicriteria Decision Making. *ICIC Express Lett.* **2022**, *16*, 145–151.
29. Qureshi, S.; Khan, S.U.R.; Javed, Y.; Saleem, S.; Iqbal, A. A Conceptual Model to Address the Communication and Coordination Challenges during Requirements Change Management in Global Software Development. *IEEE Access* **2021**, *9*, 102290–102303. [CrossRef]
30. Achimugu, P.; Selamat, A.; Ibrahim, R.; Mahrin, M.N. A systematic literature review of software requirements prioritization research. *Inf. Softw. Technol.* **2014**, *56*, 568–585. [CrossRef]