RESEARCH & INNOVATION DIVISION
IJIST

Requirements Prioritization- Modeling Through Dependency and Usability with Fusion of Artificial Intelligence Technique

Mariya Tauqeer*, Halima Jamil
University of Management and Technology (UMT), Lahore, 54600, Pakistan
***Correspondence**: mariyatauqeer870@gmail.com

Citation | Tauqeer. M, Jamil. H, "Requirements Prioritization- Modeling Through Dependency and Usability with Fusion of Artificial Intelligence Technique", IJIST, Vol. 07 Issue. 01 pp 146-158, Jan 2025
Received | Dec 26, 2024 **Revised |** Jan 17, 2025 **Accepted |** Jan 18, 2025 **Published |** Jan 19, 2025.

Abstract.
Requirements Prioritization is a crucial part of Requirements Engineering which helps to prioritize the customer's requirements according to his needs and priorities. This prioritization describes which requirements should be addressed first and which can be addressed later in the software development process. Researchers have suggested many methods and techniques of requirements prioritization. However, there is no comprehensive technique that can be used for all sizes of software projects. This research paper includes an overview of the concept of requirements prioritization, the most common techniques used to prioritize the requirements, and their comparison. Based on based on this comparison, a new requirements prioritization technique is presented in this paper which can be used for every size of a software project. This technique aims to provide the solution to many issues of previous techniques especially dependencies of requirements, user involvement as well as designers involvement. The results demonstrated that the RP model outperforms traditional techniques, particularly in agile development environments, by providing a more efficient and flexible prioritization process. The involvement of designers in requirements prioritization and handling of requirements dependencies reduces the efforts required in the design process.

Abbreviations.
Enhanced Multi-Verse Optimizer (EMVO)

Goal-oriented requirement Language (GRL)

Analytical Hierarchy Process (AHP)

Proposed Technique (RP-Model)

Keywords: Artificial Intelligence, Requirements, Prioritization, Minimax, Optimization.

Introduction:

Computers and technology have overtaken the world these days, thus the need for software products is increasing day by day. However, the failure rate of the software is not minimized to this extent. In software development, requirement engineering plays a vital role as it states the user's need and the success rate of software depends upon the fulfillment of the user's need. Requirements Engineering involves many activities; feasibility study, compilation of criteria, identification, structuring, prioritization, testing, configuration, and management [1]. Requirements prioritization is one of the most important activities in requirements engineering. It helps to prioritize the user's needs so that the most important functions can be delivered first and the others can be addressed later. In projects where resources like time and budget are limited, requirements prioritization is necessary to meet deadlines and low budgets. Competitive edge is another major business goal that can be met by delivering the most important functionalities of the software first, by using requirements prioritization. For requirements prioritization, stakeholders are actively engaged to ensure they can prioritize the most critical requirements based on their preferences and needs [2].

Literature highlights many requirements prioritization techniques, which have some gaps i.e., less user involvement, no scalability, etc. The dependencies of requirements are another important issue of these techniques. Addressing them during the early stages of software development, such as the Requirements Engineering phase, offers several benefits: ensuring the applicability of requirements, minimizing effort in the design phase, detecting errors earlier, and resolving conflicts through mutual consensus between requirements engineers and software designers [3].

This research paper proposes a new requirements prioritization technique to incorporate the above benefits. This technique involves usability, scalability, dependency, accuracy, user, and designer. In this research paper, we will also discuss the most common techniques for requirements prioritization which include: Numerical Assignment, Moscow, Binary Search Tree Technique, AHP, Bubble Sort Technique, Hundred Dollar Technique, and Planning Game [4]. We aim to analyze these techniques based on Usability, Scalability, Accuracy, and Stakeholder Involvement. Finally, an evaluation of the proposed technique is also performed in this research paper. This research paper aims to answer the following research questions:

- R1: What is the importance of requirements prioritization?
- R2: What are the major requirements prioritization techniques and their characteristics?
- R3: What can be the best technique or method for requirements prioritization?
- R4: What are the pros and cons of the proposed technique?

Objectives:

- To emphasize the importance of requirements prioritization in software development.
- To analyze major requirements prioritization techniques and their characteristics.
- To propose a new technique (RP-Model) for requirements prioritization that incorporates usability, scalability, dependency management, accuracy, and user/designer involvement.

To evaluate the proposed technique in terms of its pros and cons.

Literature Review:

Software testing is often constrained by budget and time, which may lead to certain tests being omitted during the testing of large software systems. Prioritizing the test cases is important in the software testing process. In requirements modeling, a meta-model is employed, and an instance of this meta-model is processed using the modified PageRank algorithm, which aids in prioritizing the requirements effectively [5]. A research paper [6] Proposed a method named EMVO. This method used an algorithm having three concepts White hole, Black hole, and Wormhole.

Prioritization of requirements is crucial for the successive development of software products and the reduction of risk. The current research aims to increase the accuracy of requirements and minimize the effort of the user that serves while making a decision. Various requirement prioritization techniques have been proposed in the literature; however, many of these techniques lack efficiency and exhibit low performance [7]. The efficiency of performance depends on the selection of the optimum technique. In [7], a new technique is introduced which is a genetic algorithm, which is efficient for requirement prioritization that reduces time and increases performance. This algorithm analyzes the input and takes an initial random population for requirements prioritization [8].

Stakeholder viewpoints are considered one of the most important parts of the prioritization of requirements. Many optimization problems are solved via a whale optimization algorithm. In another study [9], the whale optimization algorithm is used to prioritize software requirements. The process involves representing requirements within the search space and prioritizing them by simulating the hunting behavior of whales. The Whale Optimization Algorithm has been evaluated for prioritizing requirements of varying sizes based on runtime performance compared to the AHP technique. The results show that the Whale Optimization Algorithm outperforms AHP by 40%. Another method [10] Is based on GRL allows the stakeholders to model their requirements as well as the influence of the organizations. Requirement prioritization plays a crucial role during the elicitation phase, as implementing all requirements in a single release is often challenging. Effective prioritization is essential for successful release planning. A study [11], deploys the goal-based requirement prioritization technique, which depends on generating relative weight for requirements according to stakeholders' goals. The goal-based requirement prioritization technique is best for solving problems of time consumption and complexity [11]. In another research paper [12], explores the process of requirements prioritization, emphasizing the importance of both functional and non-functional requirements. This paper discusses the most used requirements prioritization techniques, including a critical analysis of such techniques, and proposes a model for the comparison of techniques to find the best one [12].

Prioritization of requirements is the most basic phase of the requirement engineering process that plays a very important role in delivering good software products that meet the stakeholder's needs. In a study [13], the Grey Wolf Optimization (GWO) algorithm is used as a metaheuristic technique that consists of grey wolves for requirement prioritization. The Grey Wolf Optimization (GWO) algorithm differs from other techniques due to its leadership hierarchy, which includes alpha, beta, delta, and omega wolves. When compared with the AHP technique, GWO demonstrates a 30% improvement in terms of runtime and its ability to handle larger dataset sizes.A new technique is established for user requirements prioritization that is named" URPCalc". This technique allows u s to estimate the change in user requirements and their impact in the future. URPCalc also ensures the quality and reduction of cost by using automated software to examine the requirements prioritization [14]. Another research paper [15] discusses requirements prioritization problems and challenges that are faced by requirements prioritization The authors of above stated study also proposed MDRE which is another efficient technique used for requirements prioritization. In a research paper [16], the authors analyze various techniques for requirements prioritization based on factors such as scale, complexity, scalability, and customization. The paper also includes a case study that discusses the selection of the most suitable requirements prioritization technique [16].

Requirements Prioritization and its Need:

Requirements refer to the functionalities needed by the user in a specific software product. During software development, there are often numerous requirements that cannot be addressed all at once. To manage this, requirements are prioritized [17]. Figure 1 shows the balance between resources and requirements i.e., resources are mostly limited for implementing

all of the requirements at once.

Figure 1. Balance between resources and requirements

In software development, requirements prioritization plays a vital role in many ways. If resources i.e., time and budget are limited, requirements prioritization can help in identifying the most crucial requirements that should be delivered in initial releases [18]. Requirements prioritization is necessary to first focus on the most important requirements to separate the domain requirements and business requirements. Prioritization helps manage and control different requirements.

Materials and Method:

Requirements Prioritization Methods or Techniques:

Many fundamental techniques have been designed for requirements prioritization. Some of these are discussed below:

Numerical Assignment:

In numerical assignment, different levels of prioritization are specified as high, medium, and low. These levels are then assigned to the requirements based on their priority. There can be a scale from 1 to 3, instead of high, medium, and low which is assigned to each requirement by the stakeholder. All the requirements assigned to a specific level have the same priority.

Moscow:

Moscow is quite similar to the numerical assignment. In this technique, four priority groups are presented i.e., Must have, should have, could have, and won't have. **Must have** means that the requirement must be present in the software. **Should have** meant if the requirement is implemented in the software, it would be better for the software product [19]. **Could have** shown if the requirement exists, it would be good for the software. **Won't have** represented the least priority requirements that would not be implemented in the current release.

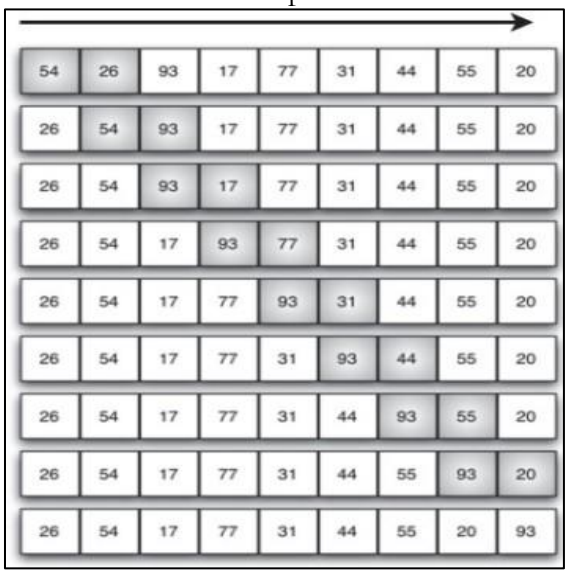

Figure 2. Bubble sort technique table [20]

Bubble Sort:

This technique originally belonged to data structures for the sorting of elements. In this technique, two requirements are compared with each other, and the requirement with high priority is selected. This selected requirement is then compared with other requirements in the same manner. In this way, a priority list of requirements is established. Figure 2 shows how the bubble sort technique is performed on different requirements.

Binary Search Tree:

This prioritization technique also originated from data structures related to the sorting of elements. This technique is applied to requirements prioritization by. In this technique, a tree is formed by using nodes. Each node has at most two child nodes. These nodes represent the requirements. First, we take a requirement and consider it as the base node, this requirement is then compared with two other requirements. The requirement with higher priority is placed on the right side while the requirement with low priority is placed on the left side. This process continues until all requirements are compared and sorted in the form of a tree.

Hundred Dollar Method:

In this technique, 100 points are given to stakeholders to divide among the requirements. The stakeholders assign the points to the requirements and the higher the points given to a requirement; the higher the requirement will be considered in priority.

Planning Game:

This technique is best suited for XP programming in agile-based projects. This technique involves stakeholders as well as developers at the same time. The stakeholders assign the requirements to three categories: Must be implemented, better to implement, and cannot implemented for the current release. These categorized requirements are then analyzed by the developers to the time taken in developing each requirement and again categorized the requirements into three groups: accurate, logically fine, and cannot be implemented.

Analytical Hierarchy Process:

Table 1. Comparison of requirements prioritization techniques

No.	Techniques	Characteristics
1	Numerical Assignment	Same priorities for the same group. Low reliability. Low fault tolerance. Not scalable. Less Accurate.
2	Moscow	Same priorities for the same group. Low reliability. Not scalable. Less Accurate.
3	Bubble Sort	Less Time-consuming. High Performance. Reliable. Efficient.
4	Binary Search Tree	Highly Scalable More Accurate. Easy to implement. Efficient.
5	Hundred Dollar Method	Inapplicable for the large number of requirements as the sum doesn't add up to 100. Hard to maintain consistency. Quick and easy for the small number of requirements.
6	Planning Game	Easy to use. Unsuitable for large projects. Same priorities for the same group.
7	AHP	More Accurate. Unsuitable for large projects because of the large number of comparisons

This technique was proposed by [12], the stakeholders break their requirements into subproblems to form a hierarchy. The developers then compare those requirements in terms of cost and time and form a comparison matrix. The recommended number of comparisons is n*(n- 1)/2 while n is the number of requirements. Table 1 shows the comparison between the above-mentioned requirements techniques.

Proposed Methodology:

The previously mentioned requirements prioritization techniques are not efficient enough to be applied across all sizes of software projects. Another important factor of requirements handling is the dependability of requirements which is not handled in any of the previously mentioned techniques efficiently.

For the solution of such problems, a technique is proposed here in which both user and designer are involved. The user prioritizes the requirements based on their preferences, while the designer manages the dependencies between the requirements. The following model provides a pictorial representation of this technique.

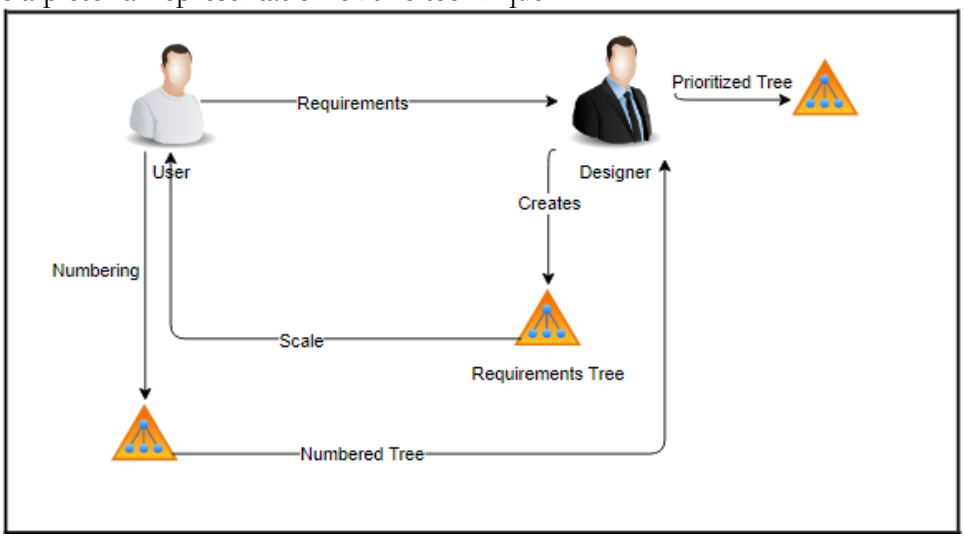

Figure 3. RP-Model

In this technique, the designer receives the requirements from the user in the layman form and arrange them in a hierarchical tree as shown in Figure 4. In this figure, R1 represents the first requirement while R2 represents the second requirement, and so on. By arranging the requirements in a hierarchical form, the dependability of requirements will be handled. Any subsequent requirement is kept under the main requirement in the subsequent levels for maintaining the hierarchy.

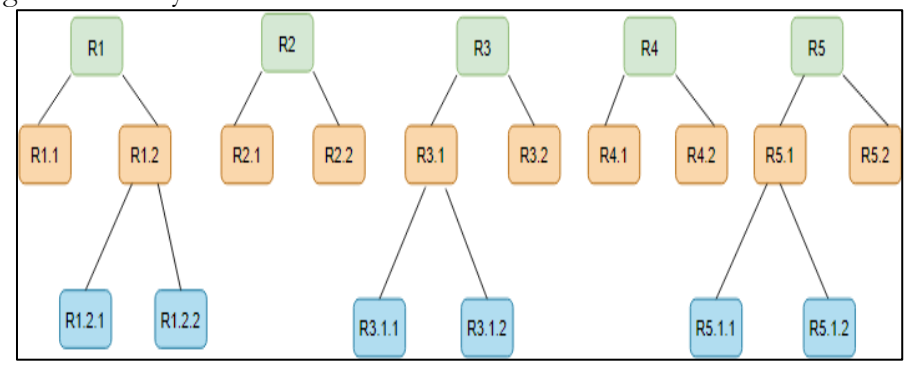

Figure 4. Requirements Tree

This requirement tree is then handed over to the user, who assigns a number to each requirement based on their preferences, as shown in Figure 5. In this figure 'P' represents the priority value along with a number given by the user. These numbers range from 1-10

representing 1 as the highest priority and 10 as the lowest priority. This scale can vary according to the size of the project.

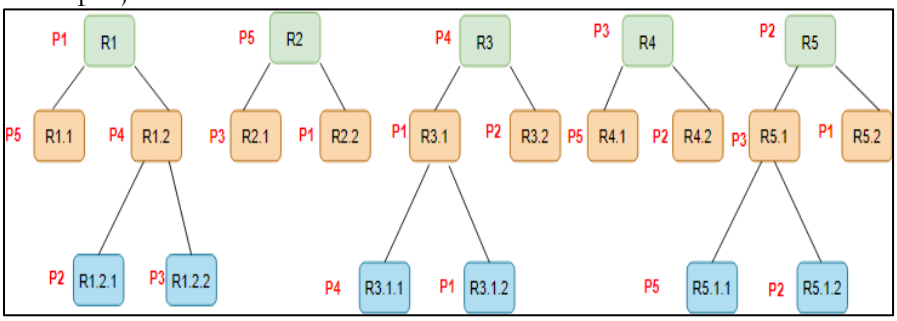

Figure 5. Numbered Tree

To make this technique efficient and time-saving, the Minimax Algorithm of Artificial Intelligence was utilized in this study. This algorithm analyzes the numbered requirements and determine which requirement should be fulfilled next. The minimax algorithm is traditionally used in a game theory to depict the next move in the game. In our technique, this algorithm assisted in efficiently selecting the next requirement to work on, as shown in Figure 6. This algorithm was highly efficient for the extraction of any number of requirements.

The scale used for prioritization here was 1 to 10. However, this algorithm could be implemented with any scale, such as 1 to 100.

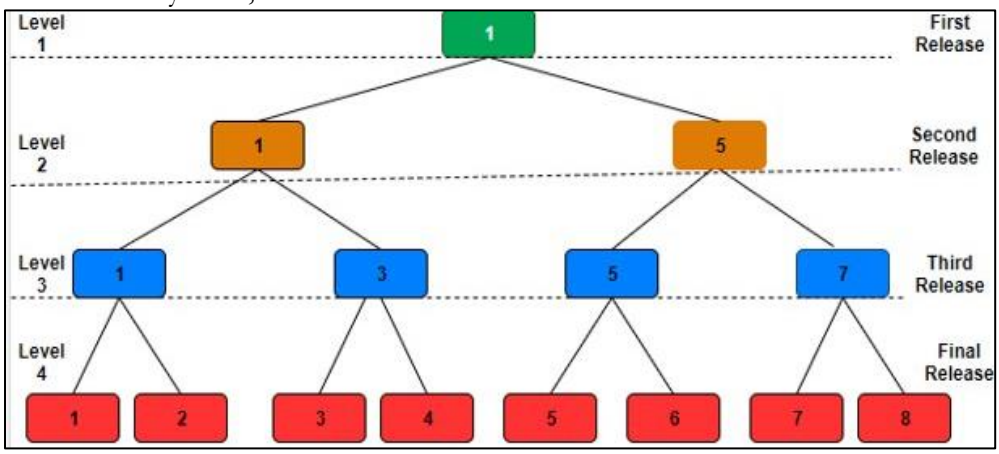

Figure 6. Minimax Algorithm

In this algorithm, initially, all numbers were at the same level. These numbers were then compared pairwise, with the lower number moving to the higher level. This process continued until all the numbers were sorted. Requirements on the highest level were implemented first and given the highest priority, with the highest priority being 1. This algorithm was applied to the numbered tree by the designer, and a final tree was formed in which all the requirements were prioritized.

Results and Discussion:

Herein this study, an email system was utilized for evaluation of proposed solution. In this system, the requirements of the user were taken and then prioritized according to our proposed technique.

Requirements:

The email system was designed to include user registration, email sending and receiving functionality, email deletion, settings options, search features, and a help menu. As it can be seen the above-mentioned requirements are ambiguous and details are hidden. To concisely manage these requirements and incorporate their dependent abilities the designer is required to form a tree known as a requirement tree based on the above requirements as shown in Figures7 and 8.

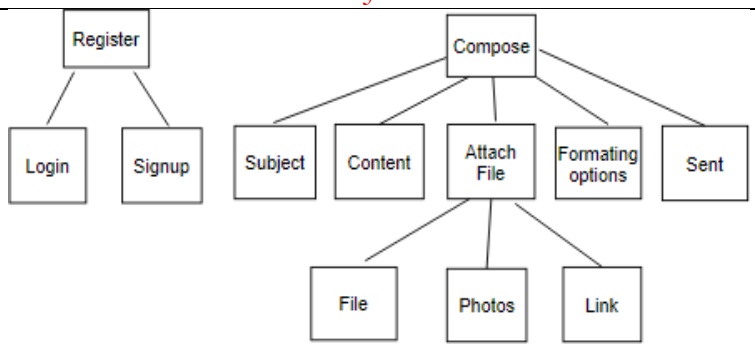

Figure 7. Requirements Tree I

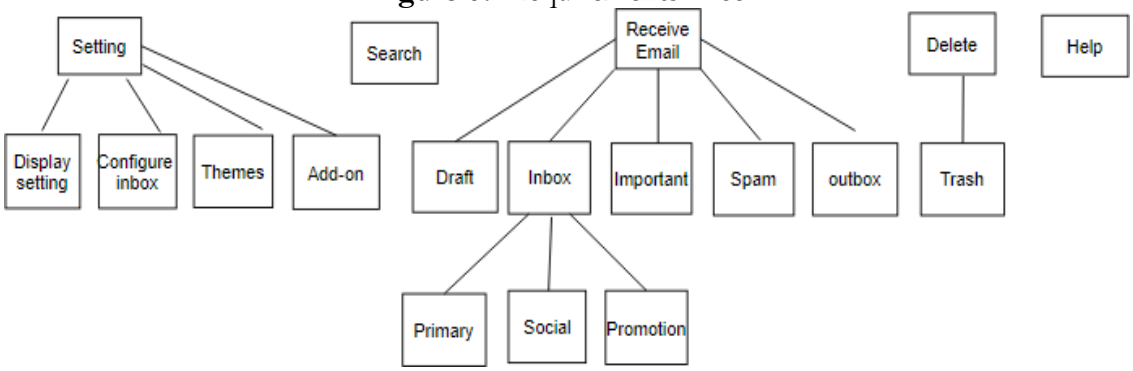

Figure 8. Requirements Tree II

The users were required to allocate numbers to the requirements based on their own preferences and priorities. The scale used by the user is 1-10 i.e., 1 represents the highest priority while 10 represents the least priority as shown in Figures 9 and 10.

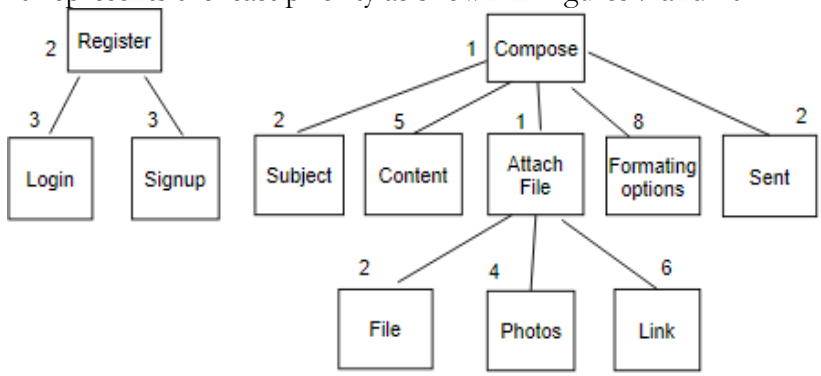

Figure 9. Numbered Tree I

Highly prioritized requirements, which were to be implemented first, were then extracted from the numbered tree using the Minimax algorithm.

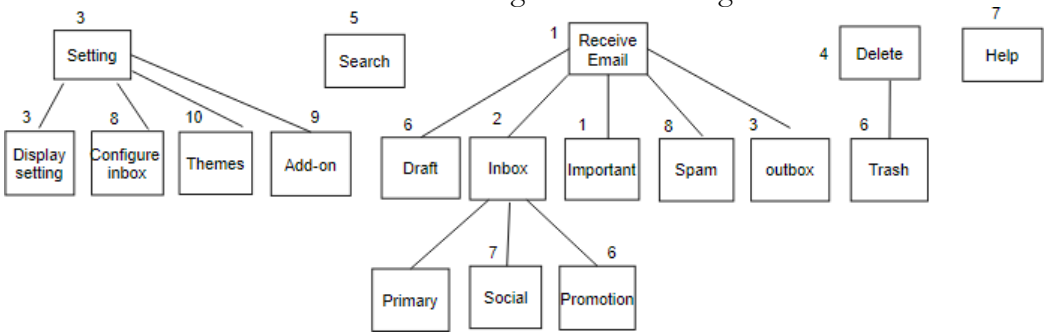

Figure 10. Numbered Tree II

The final prioritized tree after the application of the Minimax algorithm is shown in: Figure 11. In this tree, all the requirements that were prioritized as the most important

requirements i.e., numbered a 1, appear at the first level in the tree, and so on.

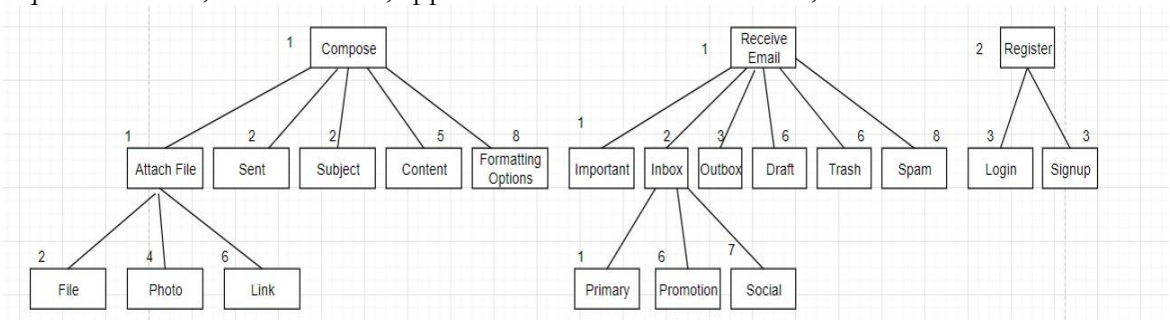

Figure 11. Prioritized Tree

The code of Minimax algorithm 12 used for prioritizing the requirements is given below in figure 12.

main.cpp

```
#include<bits/stdc++.h>
using namespace std;

int m(int depth, int nodeIndex, bool isMin,
            int scores[], int h)
{
      if (depth == h)
        return scores[nodeIndex];


    if (isMin)
        return min(m(depth+1, nodeIndex*2, true, scores, h),
            m(depth+1, nodeIndex*2 + 1, true, scores, h));
}
int log2(int n)
{
  return (n==1)? 0 : 1 + log2(n/2);
}

int main()
{
    int scores[] = {1,2,3,4,5,6,7,8,9,10};
    int n = sizeof(scores)/sizeof(scores[0]);
    int h = log2(n);
    int res = m(0, 0, true, scores, h);
    cout << "The optimal value is : " << res << endl;
    return 0;
}
```

Figure 12. Code of Minimax Algorithm

Figure 13 shows the output of the above-mentioned code. The code returns the smallest value which is the optimal value representing the most important requirement.

```
The optimal value is : 1


...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 13. Output

Comparison of Proposed Technique with Other Techniques:

In this section, the comparison of our Proposed Technique (RP-Model) with previously mentioned techniques is per- formed with respect to some parameters as shown in Table 2.

Table 2. Comparison of the Proposed technique with other techniques

Techniques	Parameters							
	Optimal Solution	Decision Making	User Involvement	Dependency	Scalability	Efficiency	Accuracy	Manageability
RP-Model	✓	✓	✓	✓	✓	✓	✓	✓
Numerical Assignment			✓					
MoScoW			✓					
Bubble Sort		✓				✓		✓
BST	✓	✓			✓	✓	✓	✓
100 Dollars			✓					
Planning Game			✓				✓	
AHP					✓		✓	✓

Pros and Cons of RP-Model:

Pros:

- The Minimax algorithm provides an optimal solution by identifying the most important requirement based on the highest priority.
- This technique will help in decision-making as we can implement the requirements that are at the highest priority in the first release.
- Direct user involvement is required in this technique which ultimately leads to user satisfaction.
- Designer is also involved throughout the process which helps to identify the dependability of requirements.
- By handling the dependencies here, will minimize the efforts required in the design phase. 6. Incorporating the dependability will result in earlier detection of errors and inapplicability of requirements. 6. This model helps in managing a large number of requirements and scales in less time and effort.

Cons:

- This model can be a little bit slow due to the branching factor of the Minimax algorithm.
- If the designer or user are not available at the time this may cause some delay in the whole process.

Discussion:

The results of this research highlight the effectiveness of the RP-Model in prioritizing requirements in complex software projects, particularly in managing dependencies and scaling with large datasets. When compared to traditional techniques such as Numerical Assignment, Moscow, Bubble Sort, and Analytical Hierarchy Process (AHP), the RP-Model demonstrated clear advantages in terms of scalability, accuracy, and user involvement [4]. Techniques like Numerical Assignment and Moscow, rely on basic categorizations, usually struggle to handle large numbers of requirements, and fail to manage dependencies, potentially leading to inefficiencies. In contrast to these methodologies, the proposed model employed a hierarchical structure that directly addresses these dependencies, providing a clearer path for

implementation. Bubble Sort is efficient in smaller datasets, but it also suffers from the same limitations and does not account for dependencies. However, the RP-Model uses the Minimax algorithm to prioritize requirements based on both their importance and interdependencies, making it more effective for larger, more complex projects [12][13].

The AHP method is highly accurate, but it becomes increasingly cumbersome as the number of requirements grows due to the need for pairwise comparisons, leading to high time complexity. The RP-Model, however, scales better with larger datasets by automating the prioritization process with the Minimax algorithm, which also incorporates designer feedback which is not present in AHP. A significant advantage of the RP-Model is its handling of requirement dependencies, an area where other methods like the Hundred Dollar and Planning Game techniques often fail or do so inefficiently. This is particularly important in avoiding conflicts during the implementation phase and ensuring smoother project development, which is a key trend in current research.

Recent trends in software engineering, highlight the needs for prioritization methods which are flexible, efficient, and capable of incorporating stakeholder feedback quickly. The RP-Model fits well within these trends, emphasizing direct user and designer involvement to make rapid decisions and adapt to changes in requirements. Additionally, the growing emphasis on automated decision-making in software engineering aligns with the RP-Model's use of artificial intelligence, particularly the Minimax algorithm, to enhance decision-making efficiency. This trend towards AI-driven requirements engineering is gaining momentum in the industry, aiming to improve software development processes by automating complex decision-making tasks, reducing human error, and ensuring greater alignment with user needs [14][15].

Despite its advantages, the RP-Model does have limitations. The Minimax algorithm, while generally efficient, may experience performance slowdowns when handling projects with a high branching factor, especially in very large datasets. Future work could focus on optimizing this algorithm to handle such cases more efficiently. Furthermore, the model's reliance on designer and user involvement may present challenges in projects with limited stakeholder availability or communication issues. Future research could also explore incorporating additional factors, such as business value, technical feasibility, and risk, into the prioritization process, making the model more comprehensive and aligning it with the growing trend of value-driven requirements prioritization, which aims to maximize the return on investment in software development projects.

Conclusion:

It is important to prioritize software requirements because without prioritization the software team can't deliver the right product according to user satisfaction at the right time. This fact may lead to software failure. There are many techniques available to prioritize software requirements i.e., Numerical Assignment, Moscow, Bubble Sort, hundred-dollar method, etc. but these techniques have some limitations. These techniques do not handle dependencies and large sets of requirements efficiently. Our proposed model involves the user alongwith the designer to prioritize requirements and efficiently help in decision-making. This model also handles dependencies by arranging the requirements in a hierarchical form. The early involvement of designers in software development leads to less effort in the design phase and also helps in the early detection of errors concerning dependencies. Hence this model can be used for efficiently managing large software requirements.

Future Work:

For future recommendations, the RP model proposed in this paper can be implemented in an automated form for prioritizing large numbers of requirements. Many other algorithms or

techniques from other knowledge areas can be used for requirements prioritization as in this research paper Minimax Algorithm is used which originated from Artificial Intelligence.

References:

[1] J. A. Ouellette and W. Wood, "Habit and Intention in Everyday Life: The Multiple Processes by Which Past Behavior Predicts Future Behavior," *Psychol. Bull.*, vol. 124, no. 1, pp. 54–74, 1998, doi: 10.1037/0033-2909.124.1.54.

[2] A. Sapunkov and T. Afanasieva, "Software for automation of user requirements prioritization," *ACM Int. Conf. Proceeding Ser.*, vol. Part F148261, pp. 1–5, 2019, doi: 10.1145/3318236.3318251.

[3] "Issues with Requirements Prioritization in MDRE." Accessed: Sep. 26, 2024. [Online]. Available: https://www.researchgate.net/publication/330533422_Issues_with_Requirements_Prioritization_in_MDRE

[4] H. Tufail, I. Qasim, M. F. Masood, S. Tanvir, and W. H. Butt, "Towards the selection of Optimum Requirements Prioritization Technique: A Comparative Analysis," *5th Int. Conf. Inf. Manag. ICIM 2019*, pp. 227–231, May 2019, doi: 10.1109/INFOMAN.2019.8714709.

[5] M. Abbas, I. Inayat, M. Saadatmand, and N. Jan, "Requirements dependencies-based test case prioritization for extra-functional properties," *Proc. - 2019 IEEE 12th Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2019*, pp. 159–163, Apr. 2019, doi: 10.1109/ICSTW.2019.00045.

[6] K. K. Adhim1, A. Hudaib2, and B. Al-Shboul3, "EFFICIENT REQUIREMENT PRIORITIZATION BASED ON ENHANCED MULTI-VERSE OPTIMIZER," *J. Theor. Appl. Inf. Technol.*, vol. 15, p. 19, 2019, Accessed: Sep. 26, 2024. [Online]. Available: www.jatit.org

[7] S. Stumpf *et al.*, "Predicting User Tasks: I Know What You're Doing!," 2005.

[8] H. Ahuja, Sujata, and U. Batra, "Performance Enhancement in Requirement Prioritization by Using Least-Squares-Based Random Genetic Algorithm," *Stud. Comput. Intell.*, vol. 713, pp. 251–263, 2018, doi: 10.1007/978-981-10-4555-4_17.

[9] A. Alzaqebah, R. Masadeh, and A. Hudaib, "Whale optimization algorithm for requirements prioritization," *2018 9th Int. Conf. Inf. Commun. Syst. ICICS 2018*, vol. 2018-January, pp. 84–89, May 2018, doi: 10.1109/IACS.2018.8355446.

[10] A. Leshob, P. Hadaya, and L. Renard, "Software Requirements Prioritization with the Goal-Oriented Requirement Language," *Lect. Notes Data Eng. Commun. Technol.*, vol. 41, pp. 187–198, 2020, doi: 10.1007/978-3-030-34986-8_13.

[11] M. A. A. Elsood, H. A. Hefny, and E. S. Nasr, "A goal-based technique for requirements prioritization," *2014 9th Int. Conf. Informatics Syst. INFOS 2014*, pp. SW18–SW24, Feb. 2015, doi: 10.1109/INFOS.2014.7036697.

[12] M. Tauqeer, S. Aslam, I. Rafique, T. Rana, and H. Jamil, "Textual vs Visual Representation- Role of Aesthetics in Human Cognition and Perception," *Innov. Comput. Rev.*, vol. 3, no. 2, p. 2023, Dec. 2023, doi: 10.32350/ICR.32.04.

[13] V. Clay, P. König, and S. König, "Eye tracking in virtual reality," *J. Eye Mov. Res.*, vol. 12, no. 1, Apr. 2019, doi: 10.16910/JEMR.12.1.3.

[14] A. Hudaib, R. Masadeh, M. H. Qasem, and A. Alzaqebah, "Requirements Prioritization Techniques Comparison," *Mod. Appl. Sci.*, vol. 12, no. 2, p. p62, Jan. 2018, doi: 10.5539/MAS.V12N2P62.

[15] R. Masadeh, A. Alzaqebah, and A. Hudaib, "Grey Wolf Algorithm for Requirements Prioritization," *Mod. Appl. Sci.*, vol. 12, no. 2, p. p54, Jan. 2018, doi: 10.5539/MAS.V12N2P54.

[16] M. Tauqeer, S. Rubab, R. A. Naqvi, and K. Javed, "Driver's emotion and behavior classification system based on Internet of Things and deep learning for Advanced Driver

Assistance System (ADAS)," *Comput. Commun.*, vol. 194, pp. 258–267, Oct. 2022, doi: 10.1016/J.COMCOM.2022.07.031.

[17] A. M. Soccini, "Gaze estimation based on head movements in virtual reality applications using deep learning," *Proc. - IEEE Virtual Real.*, pp. 413–414, Apr. 2017, doi: 10.1109/VR.2017.7892352.

[18] Q. V. Adiwardana, D., Luong, M.-T., So, D. R., Hall, J., Fiedel, N., Thoppilan, R., Yang, Z., Kulshreshtha, A., Nemade, G., Lu, Y., Le, "Advancing a conversational AI that mimics human interaction," 2020.

[19] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *Adv. Neural Inf. Process. Syst.*, vol. 2020-December, May 2020, Accessed: Oct. 01, 2023. [Online]. Available: https://arxiv.org/abs/2005.14165v4

[20] "Rehber: Bubble Sort algoritması ile sıralama | Technopat Sosyal." Accessed: Jan. 25, 2025. [Online]. Available: https://www.technopat.net/sosyal/konu/bubble-sort-algoritmasi-ile-siralama.1299566/