Ibn Al-Haitham Journal for Pure and Applied Sciences
Journal homepage: jih.uobaghdad.edu.iq
PISSN: 1609-4042, EISSN: 2521-3407
IHJPAS. 2024, 37(3)

Developing a Requirements Model for Software Projects

Haifaa Jassim Muhasin[1], Ali Yahya Gheni[2*], Hiba Adil Yousif [3] and Yusmadi Yah Jusoh[4]

[1,2,3]Department of Computer Science, College of Education for Pure Science(Ibn Al-Haitham), University of Baghdad, Baghdad, Iraq.
[4]Department of Information Systems Faculty of Computer Science and Information Technology, Universti Putra Malaysia.
*Corresponding Author.

Received:8 November 2023 Accepted:16 April 2024 Published:20 July 2024
doi.org/10.30526/37.3.3823

Abstract

Requirements elicitation is recognized as one of the most critical activities in the software development process as it has an impact on its success. Studies such as the CHAOS Report, a study based on The [Standish Group]'s CHAOS Research Project on IT project success rates and project management best practices report, indicate that about half of the factors associated with project success are related to requirements. Previous studies showed several problems related to requirements elicitation. This paper tried to find out the existing requirements models for software projects and how to develop a new requirements model for software projects. The requirements system was developed by Visual Studio 2019. Online model verification was conducted with 6 experts from the IT industry. After the system was developed, it was validated by three developers/programmers from the same industry (functional test/white-box testing) and eleven developers (non-functional test/black-box testing). The results of the model verification supported the model of requirements. Additionally, the requirements system was validated by non-functional test/black-box testing and functional test/white-box testing.

Keywords: Requirements elicitation**,** System development life cycle (SDLC)**,** Model verification**,** Model validation

1. Introduction

As it directly affects the success of the System development life cycle (SDLC), the new seven phases of SDLC include planning, analysis, design, development, testing, implementation, and maintenance. Requirements elicitation is seen as one of the most crucial processes [1]. According to Bohem [2,3], requirements elicitation is the first and most important phase in the requirements engineering process. If it is done incorrectly, low-quality products, late deliveries, and large prices will result. The Standish Group Report [4] states that there were more unsuccessful initiatives in 2006, 2008, and 2010 than there were in those years. A list of project failure-causing factors is defined in this paper. Additionally, one of the main issues with the most significant percentage (13.1%) is an incomplete requirement. The report also describes the three main reasons for project success, which are user involvement, executive management support, and a clear statement of requirements [5].

The Software Engineering Body of Knowledge (SWEBOK) [6] states that the requirements collecting, analysis, design, architecture, implementation, and maintenance phases make up the software development process. The first and most crucial step is gathering requirements [7]. Since requirements describe what the system should accomplish, the services it should provide, and the constraints on how it can operate, they reflect user demands [8]. Requirements engineering is the broad field of activities and methods used to comprehend requirements. It entails identifying the objectives, demands, and expectations of stakeholders and sharing them with the developers [9].

Loucopoulos et al. [10] define requirements Elicitation as the process of acquiring all relevant knowledge to produce a requirements model of a problem in a specific domain.

According to Borland [11], elicitation is the ability to work collaboratively with stakeholders to discover the current product needs and agree upon the vision and goals of the proposed project. According to the SWEBOK [12], this task is broken down into two activities: Requirements sources and Elicitation techniques. On the other hand, Pohl [13] defines the requirements elicitation as a core activity of the requirements engineering, which consists of:

(1) Identify sources of the relevant requirements, (2) Identify the requirements of these sources, and (3) Develop new requirements. Mulla et al. [14] define the process of requirements elicitation as follows: (1) Identify requirements sources,

(2) Collect the wish list for each corresponding part, (3) Document and Refine the wish list, (4) Integrate the wish lists with the various stakeholders, and (5) determine the non-functional requirements.

According to [4], all projects begin with a statement of requirements. Requirements are descriptions of how a software product should perform. They typically refer to some aspect of a new or enhanced product or service.

The widely cited IEEE 610.12-1990 standard [15] defines a requirement as:

(1) A condition or capability needed by a user to solve a problem or achieve an objective,

(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents, A documented representation of a condition or capability as in (1) or (2).

For a project to be successful, the criteria must be of high quality [16]. However, eliciting requirements is not a simple task. This work is complex and contradictory due to user and analyst viewpoints, mental models, and expectation mismatches. The clients' true needs are frequently not fully understood by them [17]. Others don't have current work processes that match what management wants. By introducing Athena, it is a serverless, interactive analytics service built on open-source frameworks that support open-table and file formats. Athena provides a simplified, flexible way to analyze petabytes of data where it lives. A method based on shared knowledge that builds system requirements incrementally from a narrative of user stories to the description of use cases, we attempt to address these issues. Athena is a method for cooperatively gathering requirements. It is built on group storytelling, in which participants share tales about the present and previous systems that underpin a particular activity. The stories are merged to form a single story. Stories are then transformed into scenarios and from scenarios to use cases. My solution consists of a knowledge model based on stories about the system, a collective construction method, and a tool to support interactions. We have conducted experimental analyses to show the effectiveness of the proposed approach [18].

Studies done in the past revealed several issues with needs elicitation. This one is a complicated process that involves all available information, including some familiarity with prior systems, according to Laporti et al. [17]. According to the study by Zhang et al. [19], inadequate and unclear requirements elicitation was one of the reasons projects failed, along with the wrong project scope. According to Mulla et al. [14], the work of eliciting requirements is challenging, particularly in large software projects with an abundance of stakeholders and information overload. However, the existing methods are not suitable for large projects [20]. Atladottir et al. [21] argue that considering users as a primary source of information leads to a favorable product. Whereby Meth et al. [22] argue that "Automation" is at the top of the wish list of most software developers and that "Identifying user needs" is not performed efficiently.

As a result, numerous literature reviews have been conducted in light of the significance of the impact of requirements elicitation on the success of software projects, including the work of Pacheco et al. [23], who reviewed methods to identify stakeholders, Carrillo et al. [24], and Meth et al. [22], who reviewed tools supports the requirements elicitation process.

The following is how the paper is structured: The first section is an introduction, which includes the related work as documented in the literature. The research technique and process are presented in section 2; the main contribution of this paper is to develop a new requirements model for software projects. The requirements system was developed by Visual Studio 2019. The system development is presented in section 3, and the results and discussions are explored in section 4. Section 5 concludes with a summary of the findings and recommendations for future research.

2. Research Method

The stages were modified to meet the research issues posed in the study. The method for choosing the papers was to review the literature, which listed the articles used in this study. The second stage involved identifying the theoretical frameworks that had been used in the literature to establish the current requirements model for software projects. In the third stage, the models that already existed were filtered before being used for classification in the fourth stage.

Figure 1 shows how the procedure for choosing papers involved leveraging five databases to compile studies on requirements models for software projects. The selection process was carried out in accordance with related advancements in the literature. To gather the research papers used for this study, it was necessary to use the top internet databases (Scopus, Science Direct, IEEE Xplore, Springer, and Emerald).

We filed the research papers according to qualification criteria, such as publication period and the context of the paper.

Because of the selection approach and context they followed, the researchers were able to locate numerous study publications without encountering the problem of duplication. This can help academic scholars understand the significance of high-quality papers even more. For example, papers that did not meet the eligibility requirements were not given further consideration.

These articles include reports, white papers, survey studies, short papers, and papers for conferences. **Figure 2** provides a more detailed description of the search and selection procedure and the query used during it.

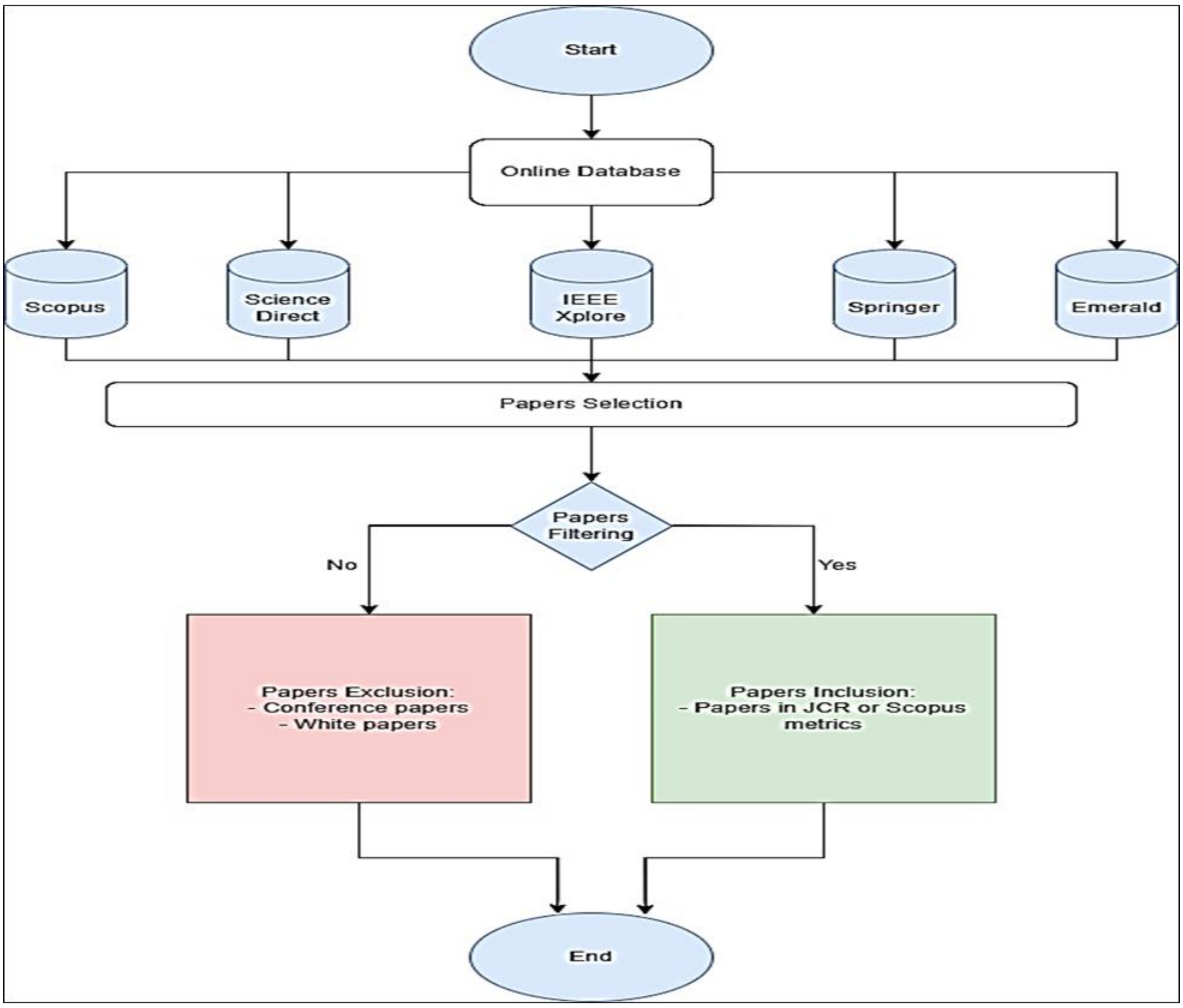


Figure 1. Paper selection methodology

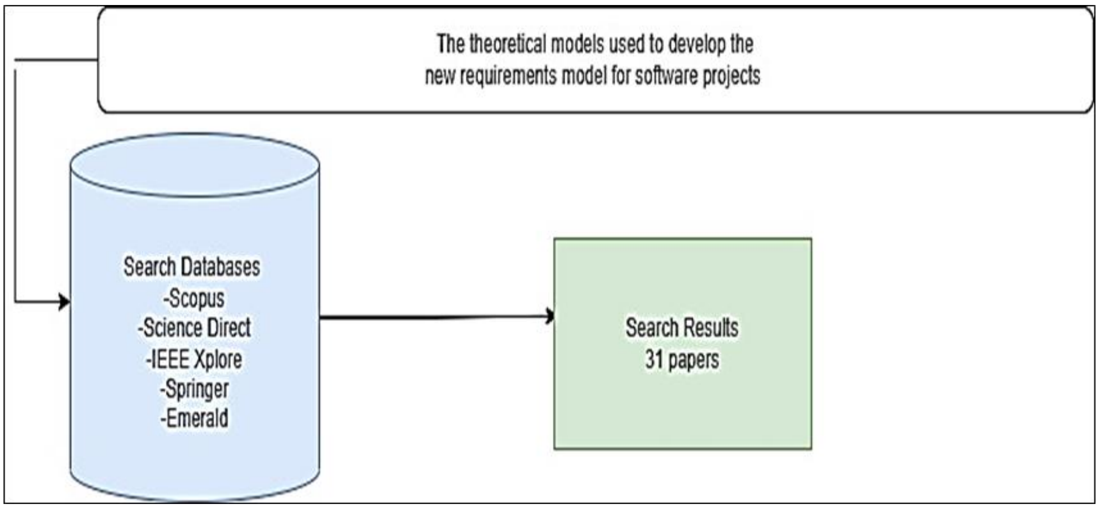


Figure 2. Papers search results

2.1. Requirement model verification

Six IT industry professionals conducted a professional evaluation of the model to confirm its validity. According to [25], three expert reviews are the required minimum. Two experts received a draft of the expert review (the first one in English and the second in questionnaire form). The expert evaluation was finished, and the findings were reviewed. Expert input enhanced the model.

2.2. Requirement model validation

Black-box testing, also known as non-functional testing, is carried out by software companies to evaluate a system's utility and usability (can I operate it?), as well as its dependability (is it consistently accurate? and how long it takes to fix), performance (does it meet its constraints with regard to response time or space requirements), robustness (does the system inform the user with a message if data do not comply to what was expected?), and correctness (does it do what I want?) After using the Requirements system, 11 developers were given a questionnaire [26,27]. Some of the questionnaires' questions, which were modified from [28] and originally from [29], questioned developers on how simple it was to complete activities using the system. The survey uses a Likert scale as its foundation (strongly agree, agree, neither agree or disagree, disagree, strongly disagree).
The assessment measures the system's utility/usability, reliability, resilience, performance, and correctness using 10 questions and a five-point Likert-type rating scale (ranging from "strongly disagree" to "strongly agree"). In the same organization, three other developers/programmers examine the system's functionality through "white-box" testing, which involves looking at the system's implementation-related code, internal logic, conditions, and loop structures.

3. Requirement Model Design

This system was created with Visual Studio 2019. The essential elements that will influence how the requirements system and user interact are entering the requirements system, entering project-related keywords, selecting the associated project, and then departing the system.

3.1. The component diagram

Models are defined by the unified modeling language (UML), which includes analysis, design, and implementation models. However, you are not required to establish or keep up with three models for a single application. A component diagram is an example of a diagram you might find in an implementation model; the component diagram is a diagram used in UML. Here, the author uses this diagram as the others did in other research papers [30,31]. An illustration of the component parts is a component diagram (think symbols). A component schematic is shown in **Figure 3**. It is crucial to list all necessary system users together with the duties they carry out.

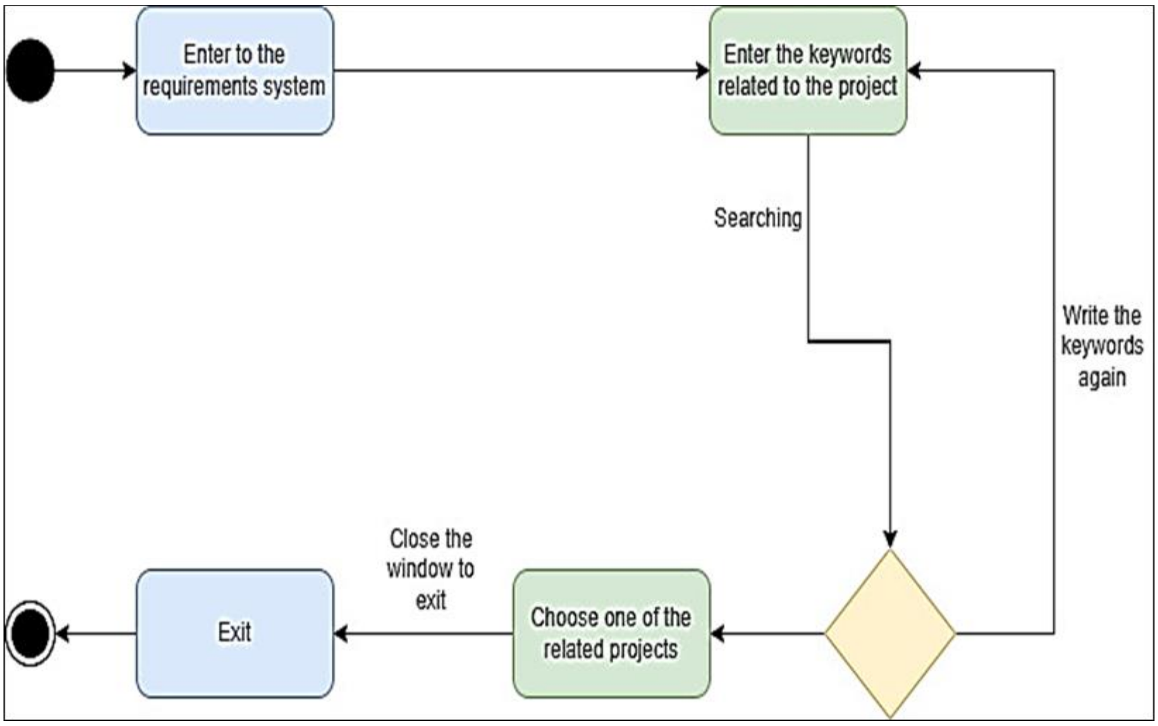

Enter to the requirements system
Enter the keywords related to the project
Searching
Write the keywords again
Close the window to exit
Exit
Choose one of the related projects

Figure 3. Requirements system component diagram

3.2. The sequence diagrams

A sequence diagram, which models a single flow across the system's objects, displays the classes along the top and the messages sent between those classes [30]. It shows the communications that take place between users and requirements management systems. By following the order of the messages from top left to bottom right, as illustrated above in **Figure 4**, a sequence diagram suggests a temporal ordering [31].

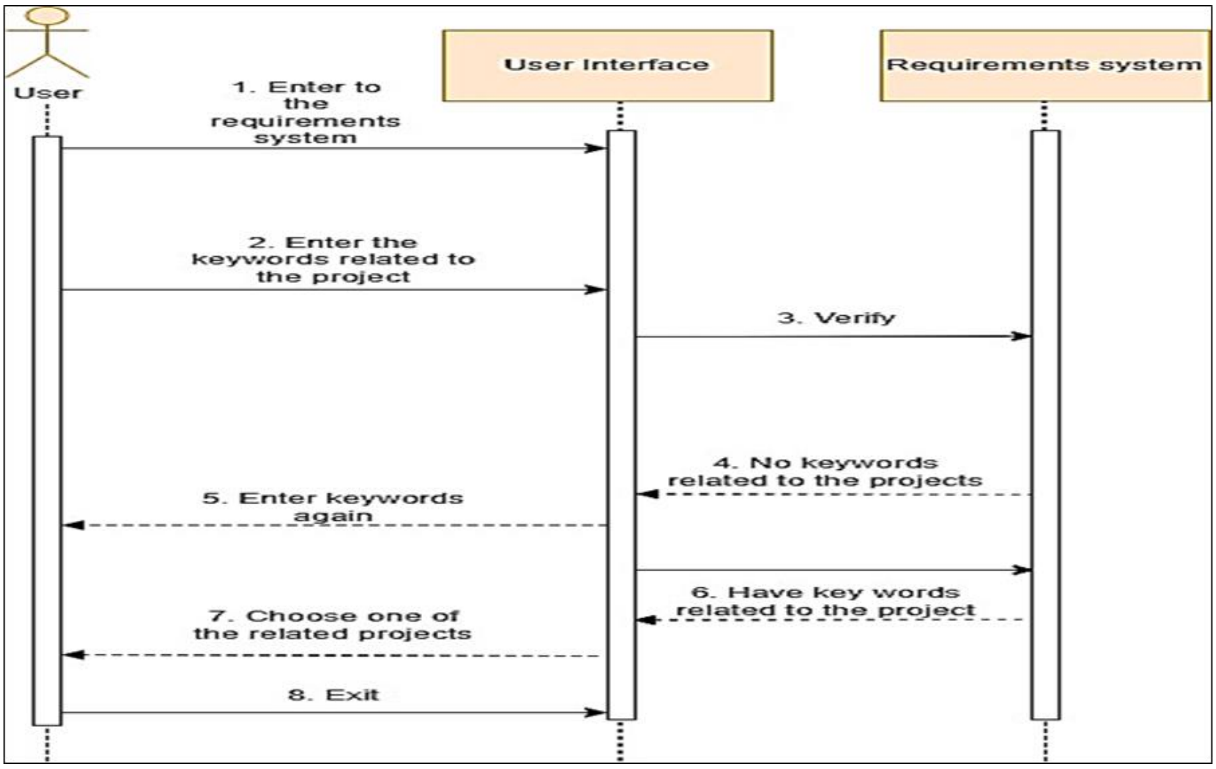
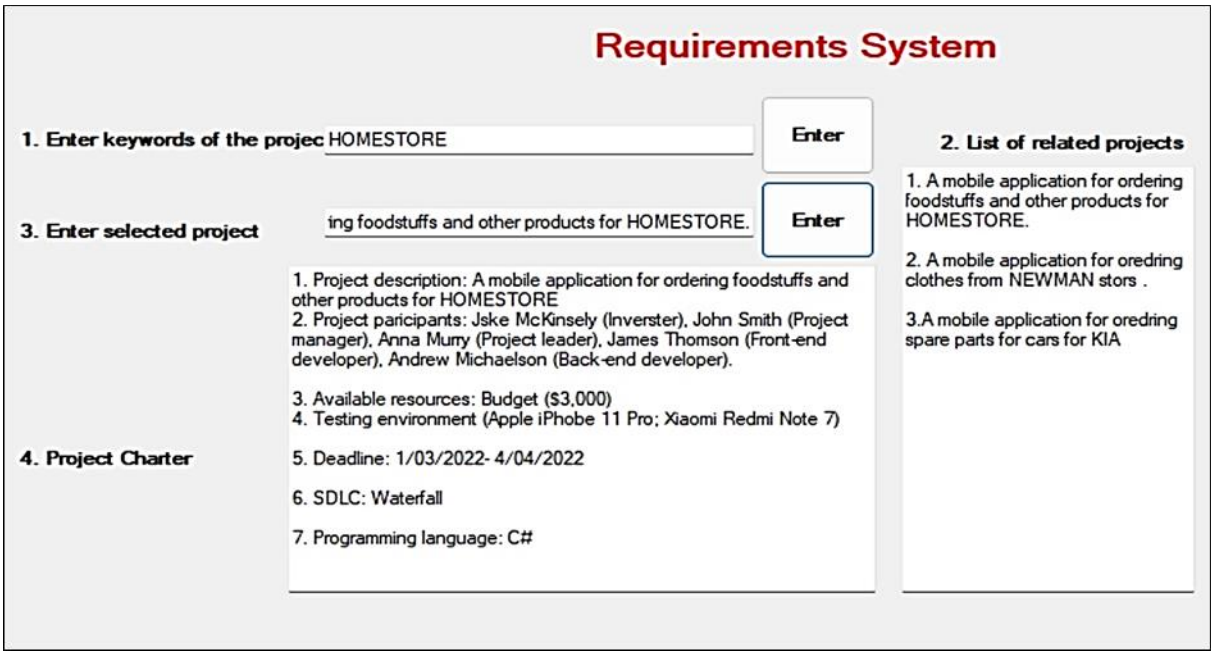

User
User Interface
Requirements system
1. Enter to the requirements system
2. Enter the keywords related to the project
3. Verify
4. No keywords related to the projects
5. Enter keywords again
6. Have key words related to the project
7. Choose one of the related projects
8. Exit

Figure 4. Requirements system sequence diagram

4. Requirement Model Implementation

The analysis and design processes determined which shape and design was chosen and implemented for the requirements system, which is available in a range of sizes and configurations. Users were asked to test out the system and give input on accessibility testing so that they could provide reasonable assistance. **Figure 5** shows a requirements system user interface and a requirements system sequence diagram.


Requirements System
1. Enter keywords of the projec
Enter
3. Enter selected project
Enter
4. Project Charter
2. List of related projects

HOMESTORE

ing foodstuffs and other products for HOMESTORE.

1. Project description: A mobile application for ordering foodstuffs and other products for HOMESTORE
2. Project paricipants: Jske McKinsely (Inverster), John Smith (Project manager), Anna Murry (Project leader), James Thomson (Front-end developer), Andrew Michaelson (Back-end developer).

3. Available resources: Budget ($3,000)
4. Testing environment (Apple iPhobe 11 Pro; Xiaomi Redmi Note 7)

5. Deadline: 1/03/2022- 4/04/2022

6. SDLC: Waterfall

7. Programming language: C#

1. A mobile application for ordering foodstuffs and other products for HOMESTORE.

2. A mobile application for oredring clothes from NEWMAN stors .

3.A mobile application for oredring spare parts for cars for KIA

Figure 5. Requirements system user interface

5. Results And Discussion

5.1. Results of requirement model verification

The draft of an expert questionnaire interview was amended after being forwarded to two experts in questionnaire design and the English language. Six IT sector professionals were sent the model and the expert questionnaire interview for verification. The results were studied. The data support experts' opinion that the proposed paradigm is applicable. The experts also concur that the suggested model is applicable, thorough, understandable, accurate, and coherent.

5.2. Results of requirement validation

Non-functional/black-box testing is carried out at a software company. After using the requirements system, 11 developers were asked to complete an online survey. The facts discovered. After reviewing the data, 81.2 percent of respondents said they would want to use the requirements system in response to item 1, which is related to usability. Item 2, with 81.2 percent of respondents strongly agreeing, states that the requirements system is correct and takes little time to repair, which is related to system efficiency." Because this is a negative issue, users' responses to "strongly disagree" (81.2%) and "disagree" (18.8%) are high ("The requirements system does not provide the user with a message if data do not comply to what was intended"). “Strongly disagree” (81.8 percent) and “disagree” (81.2 percent) were similarly used by respondents for question 6, which is related to system correctness (“The requirements system do not do what I want”) (18.8 percent).

In conclusion, does the system notify the user when the data does not match what was anticipated. Users commended the robustness of the system. The system satisfies its reaction time or space demands limits, as shown in Item 10 of the list. The system is useful/usable, dependable, resilient, performs well, and is correctness, according to the results of the ten non-functional test/black-boxing aspects. The functional test/white-box testing was carried out by three developers/programmers from the same company, who verified the code, the software's basic logic, conditions, and loop structures.

6. Conclusion

A literature review was lastly conducted to underline the requirements and challenges associated with software projects. The requirements system was also created to raise the success rate of software projects. Literature reviews have been conducted considering the significance of the impact of requirements elicitation on the success of software projects. Finally, the requirements system was developed by Visual Studio 2019. More research is required to add functionality to the requirements management system and add more features to the existing system.

Acknowledgements

First, we would like to express our sincere gratitude to the University of Baghdad for continuously supporting research in my academic environment. Second, many thanks to Universit Putra Malaysia (UPM) for all the knowledge we gained during my Ph.D. and research journey.

Conflicts of Interest

The authors declare no conflict of interest.

Funding

Non

References

1. Maznevski, M.L.; Chui, C. Leading global teams. In *Global leadership*; Routledge. **2017**, 273-301. https://doi.org/10.4324/9781315232904.
2. Marchewka, J.T. *Information technology project management: Providing measurable organizational value*; John Wiley & Sons. **2016**, ISBN 13:978-1118911013.
3. Boehm, B.; Grunbacher, P.; Briggs, R.O. Developing groupware for requirements negotiation: lessons learned. *IEEE Software*. **2001**, *18(3)*, 46-55. https://doi.org/10.1109/52.922725.
4. Hussain, A.; Mkpojiogu, E.O.; Kamal, F.M. The role of requirements in the success or failure of software projects. *International Review of Management and Marketing*. **2016**, *6(7)*, 306-311. https://doi.org/306 - 311, 01.08.2016.
5. Savolainen, P.; Ahonen, J.J.; Richardson, I. Software development project success and failure from the supplier's perspective: A systematic literature review. *International Journal of Project Management*. **2012**, *30(4)*, 458-469. https://doi.org/10.1016/j.ijproman.2011.07.002.
6. Sedelmaier, Y.; Landes, D. Software engineering body of skills (SWEBOS). In *Proceedings of the 2014 IEEE Global Engineering Education Conference (EDUCON)*. **2014**, 395-401. https://doi.org/10.1109/EDUCON.2014.6826125.
7. Gillani, M.; Niaz, H.A.; Ullah, A. Integration of software architecture in requirements elicitation for rapid software development. *IEEE Access* **2022**, *10,* 56158-56178. https://doi.org/10.1109/ACCESS.2022.3177659.
8. Kasauli, R.; Knauss, E.; Horkoff, J.; Liebel, G.; de Oliveira Neto, F.G. Requirements engineering challenges and practices in large-scale agile system development. *Journal of Systems and Software*. **2021**, *172*, 110851. https://doi.org/10.1016/j.jss.2020.110851.
9. Nuseibeh, B.; Easterbrook, S. Requirements engineering: a roadmap. In *Proceedings of the Proceedings of the Conference on the Future of Software Engineering*. **2000**, 35-46. https://doi.org/10.1145/336512.336523.
10. Loucopoulos, P.; Karakostas, V. *System requirements engineering*; McGraw-Hill, Inc. **1995**; ISBN 978-0-07-707843-0.
11. Coplien, J.O. Borland software craftsmanship: A new look at process, quality and productivity. In *Proceedings of the 5th Annual Borland International Conference*. **1994**, *5*.
12. Abran, A.; Moore, J.W.; Bourque, P.; Dupuis, R.; Tripp, L. Software engineering body of knowledge. *IEEE Computer Society, Angela Burgess*. **2004**, *25*, 1235, ISBN 0-7695-2330-7.
13. Pohl, K. *Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant*; Rocky Nook, Inc. **2016**, ISBN 978-1-937538-77-4.
14. Mulla, N.; Girase, S. A new approach to requirement elicitation based on stakeholder recommendation and collaborative filtering. *International Journal of Software Engineering & Applications*. **2012**, *3(3)*, 51. https://doi.org/10.5121/ijsea.2012.3305.
15. Wahono, R.S. Analyzing requirements engineering problems. In Proceedings of the IECI Japan Workshop, **2003**.
16. Frefer, A.; Mahmoud, M.; Haleema, H.; Almamlook, R. Overview success criteria and critical success factors in project management. *Industrial engineering & management*. **2018**, *7(1)*, 1-6. https://doi.org/10.4172/2169-0316.1000244.
17. Laporti, V.; Borges, M.R.; Braganholo, V. Athena: A collaborative approach to requirements elicitation. *Computers in Industry*. **2009**, *60(6)*, 367-380. https://doi.org/10.1109/CSCWD.2007.4281527.
18. Sommerville, I. *Software Engineering, 9/E*; Pearson Education India **2011**.
19. Zhang, Y.; Harman, M.; Finkelstein, A.; Mansouri, S.A. Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation. *Information and software technology*. **2011**, *53(7)*, 761-773. https://doi.org/10.1016/j.infsof.2011.02.001.

20. Krane, H.P.; Rolstadås, A.; Olsson, N.O. Categorizing risks in seven large projects—Which risks do the projects focus on? *Project management journal*. **2010**, *41(1)*, 81-86. https://doi.org/10.1002/pmj.20154.
21. Atladottir, G.; Hvannberg, E.T.; Gunnarsdottir, S. Comparing task practicing and prototype fidelities when applying scenario acting to elicit requirements. *Requirements Engineering*. **2012**, *17*, 157-170. https://doi.org/10.1007/s00766-011-0131-2.
22. Meth, H.; Brhel, M.; Maedche, A. The state of the art in automated requirements elicitation. *Information and Software Technology*. **2013**, *55(10)*, 1695-1709. https://doi.org/10.1016/j.infsof.2013.03.008.
23. Pacheco, C.; Garcia, I. A systematic literature review of stakeholder identification methods in requirements elicitation. *Journal of Systems and Software*. **2012**, *85(9)*, 2171-2181. https://doi.org/10.1016/j.jss.2012.04.075.
24. De Gea, J.M.C.; Nicolás, J.; Alemán, J.L.F.; Toval, A.; Ebert, C.; Vizcaíno, A. Requirements engineering tools: Capabilities, survey and assessment. *Information and Software Technology*. **2012**, *54(10)*, 1142-1157. https://doi.org/10.1016/j.infsof.2012.04.005.
25. Asarani, N.A.M.; Ab Rahim, N.Z. Preliminary study of online training implementation from multiple perspectives in malaysian public sectors. *Journal of Theoretical and Applied Information Technology* **2016**, *90(1)*, 77.
26. Albert, B.; Tullis, T. *Measuring the user experience: collecting, analyzing, and presenting usability metrics*; Newnes: **2013**, ISBN 978-0124157811.
27. Gheni, A.Y.; Yousif, H.A.; Jusoh, Y.Y. Online medical consultation: covid-19 system using software object-oriented approach. *Bulletin of Electrical Engineering and Informatics*. **2021**, *10(6)*, 3471-3480. https://doi.org/10.11591/eei.v10i6.3189.
28. Harrati, N.; Bouchrika, I.; Tari, A.; Ladjailia, A. Exploring user satisfaction for e-learning systems via usage-based metrics and system usability scale analysis. *Computers in Human Behavior*. **2016**, *61*, 463-471. https://doi.org/10.1016/j.chb.2016.03.051.
29. Brooke, J. SUS-A quick and dirty usability scale. *Usability evaluation in industry*. **1996**, 189-194, 4-7, ISBN 9780429157011.
30. Lano, K. *Advanced systems design with Java, UML and MDA*; *Elsevier:* **2005**; ISBN 9780080456911.
31. Gemino, A.; Parker, D. Use case diagrams in support of use case modeling: Deriving understanding from the picture. *Journal of Database Management (JDM).* **2009**, *20(1)*, 1-24. https://doi.org/10.4018/jdm.2009010101.