International Journal on Perceptive and Cognitive Computing (IJPCC) Vol 10, Issue 1 (2024)
https://doi.org/10.31436/ijpcc.v10i1.456

Collaborative Requirements Review Tool (Collaborev) to Support Requirements Validation

Jamaludin, Nazurin, Ahmad Kamran, Alya, and Nordin, Azlin*

Department of Computer Science (DCS), Kulliyyah of Information and Communication Technology (KICT), International Islamic University Malaysia (IIUM), 53100 Gombak Kuala Lumpur, Malaysia,

*Corresponding author: azlinnordin@iium.edu.my
(Received: 24th November 2023; Accepted: 30th December 2023; Published on-line: 28th January 2024)

Abstract— Requirements review involves a team of reviewers, who will go through the requirements in an attempt to find any requirements issues. The checklist-based technique is the most typically adopted method during requirements reviews. Due to the inconsistency and inefficiency of manual requirements review procedures, requirement engineers often find it challenging to effectively and productively review requirements. By streamlining the process with automated processes, collaborative features, and clear traceability, the web-based solution increases productivity, accuracy, and transparency of review. This innovative solution solves the shortcomings of manual procedures, paving the way for more fruitful software development endeavours. The software project was planned and managed using the Model-View-Controller (MVC) architecture framework and adopted an iterative prototyping methodology. The implementation of the project is using Laravel framework by utilizing PHP, JavaScript, HTML, CSS, Bootstrap, and MySQL.

Keywords— requirements validation, requirements error, requirements review, reading techniques, checklist-based technique

I. INTRODUCTION

Requirements engineering (RE) is a critical phase in software development that focuses on understanding, documenting, and managing the needs and expectations of stakeholders. RE activities contribute to ensuring the success of software development projects by establishing a clear and shared understanding of what the software should do and how it should behave. The RE process focuses on the production and refinement of a software requirements specification (SRS) or system-specific group of requirements [1].

The impact of RE in software development has been stated in many literatures including in [2]. Many researchers have proposed quite a number of RE methods from different perspectives including goal-based, model-based, scenario-based, and value-based RE methods to achieve quality requirements and reduce development efforts to ensure the success of projects [3].

The majority of RE approaches generally involve five primary activities that are elicitation, analysis and negotiations, documentation, validation, and (5) management [4]. IEEE Standard for Software Review and Audit lists five requirements validation techniques i.e. management review, technical review, inspection, walkthrough, and audit During the requirements review process, the stakeholders examine the requirements, look for any problems, compile and discuss the problems, and decide on a course of action to fix the problems [5]. The purpose of a technical review is for an expert group of individuals to assess a software product to determine its fitness for its intended use and identify any deviations from standards and specifications. The review is not only specific to RE phase, but it can be applicable to any stage of the software development life cycle, such as design, and implementation. For example, in a design review, the reviewers review the design documents and look for any potential issues.

A review process is also considered as a learning and sharing knowledge instead of focusing on the main objective of the review process, which is finding faults [7]. Nonetheless, the challenges of achieving good validation and verification quality is addressed by practices to ensure clear and agreed requirements [6]. The conventional method of conducting requirements review sessions has shortcomings, notwithstanding the significance of requirements reviews. It could require a lot of time and effort for the organizer to manually schedule time and confirm each reviewer's availability. The availability of each reviewer could also change as a result of other commitments.

In addition, the conventional requirements review process is often difficult to manage and track, especially when dealing with large and complex requirements documents, time-consuming and prone to inconsistencies [7], which can lead to errors and delays in the development process. Conventional manual reviews are often plagued by inefficiencies, inconsistencies, and

https://doi.org/10.31436/ijpcc.v10i1.456

limitations in collaboration, leading to project delays, inflated costs, and compromised software quality.

In the ever-evolving landscape of software development, rigorous and efficient requirements review play a critical role in ensuring project success. Recognizing these shortcomings in the conventional requirements review process, this project embarks on a mission to develop a web-based system specifically designed to automate the requirements review process by implementing the following goal: To develop a web-based system to streamline the requirements review process by leveraging web technologies to establish an efficient and user-friendly platform that enhances the overall workflow of environment. This collaborative enhancement has the potential to reduce miscommunications, improve decision-making, and align the entire team towards a unified vision, thereby expediting the entire software development lifecycle requirements validation.

Collaborative Requirements Review Tool (Collaborev), is a project that revolves around enhancing existing web-based tools that are used in the requirements review process. The motivation of this project started with work in [8], and the enhancement of the work had been conducted and published in [7], [9]. However, Collaborev is not a mere enhancement but an entirely built from scratch from those initial ideas. The project commenced with extensive research, including a literature review and comparative analysis of existing requirements review tools.

This system is strategically engineered to streamline collaboration among document authors, review leaders, and reviewers. With a user-centric interface and a focus on checklist-based reading techniques, Collaborev aims to transcend the limitations of the conventional requirements review process by addressing the following objectives:

1. To develop a web-based system to streamline the requirements review process by leveraging web technologies.
2. To implement customized checklist-based reading technique.
3. To facilitate collaboration between authors, reviewers, and review leaders through a shared platform to enhance teamwork, fostering a more efficient and interconnected review environment.
4. To generate detailed reports summarizing review findings, decisions, and changes.
5. To improve the efficiency and accuracy of requirements review.
6. To evaluate the tool to ensure that the requirements and objectives of the project are met.

This collaborative enhancement has the potential to reduce miscommunications, improve decision-making, and align the entire team towards a unified vision, thereby expediting the entire software development lifecycle requirements validation. Meanwhile, the consistent and thorough reviews enhance the reliability of the review process, addressing potential oversights and promoting a comprehensive evaluation of requirements.

Within the requirements engineering (RE) context, Collaborev is significant in many ways and has an enormous effect on how the discipline develops. With conventional requirements review processes being laborious and prone to human mistake, review teams using labor-intensive approaches run the risk of misinterpreting or accidentally omitting important elements. This project aims to alleviate this significant challenge. The goal of Collaborev to reduce human error and increasing software product quality during this crucial stage of development can be achieved by implementing an organized web-based system. This effectiveness decreases rework during later phases of development and increases customer satisfaction by guaranteeing that the final products more closely align with stakeholder needs while also saving significant time [1].

Furthermore, Collaborev serves as a catalyst for improving collaboration between the various parties being involved in the requirements review process. A collaborative environment may be established more easily due to the platform's straightforward user interface and clearly defined roles between document authors, review leaders, and the reviewers. This collaborative improvement can facilitate better decision-making, help to recognize and resolve misunderstandings, and streamline the team as a whole in the direction of a focused, cohesive goal that will substantially shorten the software development life cycle.

Improved requirements validation methods could lead to better software quality, which would benefit customers, end users, and product owners. They would also find it beneficial to have a thorough evaluation report so they could assess the requirements' quality and take the necessary measures.

Lastly, Collaborev is relevant not only because it can revolutionize requirements review process but also because it could promote technological innovation, efficiency, and collaboration in software engineering (SE), which will eventually lead to simplified procedures. Stakeholders such as product owners, clients, and end-users could benefit from the improved quality of the software resulting from the more efficient and effective requirements validation process. They could also benefit from the comprehensive review report and visualization features that would allow them to better understand the quality of the requirements and take appropriate action to address issues.

https://doi.org/10.31436/ijpcc.v10i1.456

In the long run, Collaborev's significance lies in its potential to reshape the requirements review process, improve efficiency and collaboration, and utilize technological innovation within SE while laying the groundwork for more streamlined and effective RE practices in the future.

II. Review of Previous Work

An investigation on previous work and existing similar tools was conducted to analyze the relevant features for the tool enhancement.

A. eReview

eReview [7], [10] is a web-based tool for requirements review, which serves as the foundation for this project. See Fig. 1. The significant strength of eReview is that it provides the option to apply a combination of two reading techniques i.e. the checklist-based and perspective- based techniques. This feature grants reviewers the flexibility to select the technique that best suits their preferences and requirements. Additionally, eReview also supports collaborative review of Software Requirements Specifications (SRS).

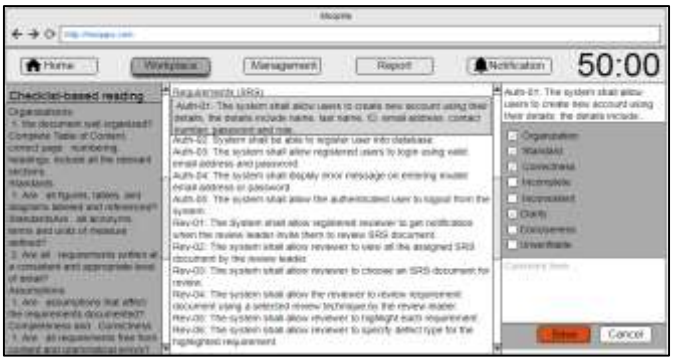

Fig. 1 eReview [7],[10]

Furthermore, a major drawback is the lack of a function that lets users monitor the progress of the documents they have evaluated. Users need this functionality to stay updated about the status of their reviews and to track the progress of their work. Users would have the required visibility and control over their review process if this feature were implemented.

B. Automated Requirements Measurement (ARM) tool

The ARM tool evaluates natural language requirements (NLR) during the early software development life cycle. It was created by the National Aeronautics and Space Administration (NASA) Software Assurance Technology Center (SATC) but has been improved in 2014 [1]. The tool searches for each of the quality primitives in the requirements document that was established by SATC that are completeness, correctness, ranked, unambiguous, consistent, customizable, traceable, and verifiable. In [14], the authors stated that the implementation of automation tool in requirements defects detection is scarce and many still rely on manual way of conducting reviews. However, these automated tools rely on the automation by the tool itself rather than being built to complement the requirements review activity, and requirements reading techniques.

B. Innoslate

The subsequent work to be investigated is Innoslate [11]. Innoslate is a robust system engineering and modelling tool encompassing a spectrum of features. While its primary focus lies in systems engineering and modelling, it doubles as collaborative requirements review tool. Its strengths include comprehensive functionality, offering an all-in-one solution for complex system development projects. Innoslate fosters collaboration through a shared platform, enabling real-time teamwork and communication within the development team. The system further provides features for generating reports and documentation, facilitating review and approval processes. See Fig.2.

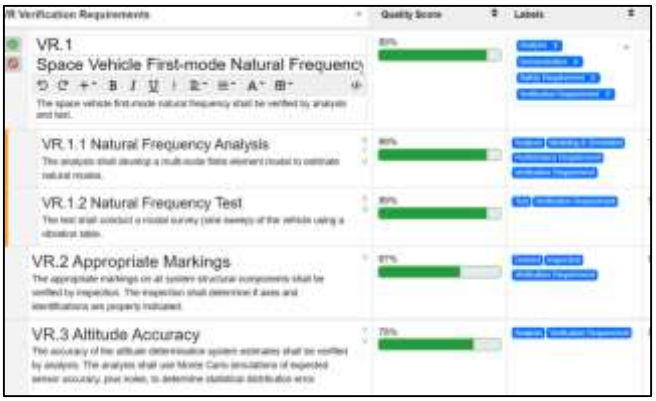

Fig. 2 Innoslate [11]

Particularly, Innoslate supports the development of visual models for system architectural procedures and requirements and is in line with Model-Based Systems Engineering (MBSE) [5]. This facilitates the analysis and comprehension of system complexity. Additionally, it helps with impact analysis and traceability, allowing users to monitor and record the connections between various system components and evaluate the effects of system component modifications. This approach does, however, have a number of shortcomings. The high learning curve is the first flaw. For new users, Innoslate may have a steep learning curve because of its extensive features. It takes a

https://doi.org/10.31436/ijpcc.v10i1.456

lot of time and work to completely comprehend and utilize the tool's possibilities.

The restriction on personalization is the second flaw. Although Innoslate has comprehensive requirements management features, there can be restrictions on how much it can be customized. Due to potential limitations or unavailability of some customization options, users may encounter difficulties tailoring the application to their unique requirements or workflows.

C. Helix ALM

The product under consideration is Helix application lifecycle management (ALM), a comprehensive system that aids teams in overseeing and controlling the whole software development process [12]. As part of the analysis, the characteristics and requirements of this system have been observed. Helix ALM offers collaborative tools that support requirements review and team member cooperation. These components ensure that individuals involved can contribute feedback, approvals, and comments at every level of the requirements review cycle, in an effort to improve the requirements review process.

Fig. 3 Helix ALM[12]

The system's primary strength is its centralized requirements repository. Helix centralized store requirements for Block ALM. This ensures uniformity and facilitates collaboration by enabling all stakeholders to access and review the criteria from a single location. It also provides review notifications and workflows. Helix ALM workflows can be customized to define the requirements review procedure. These processes could include multiple steps, such as collecting feedback, conducting a preliminary review, and granting final approval. It is also possible to set up these alerts and reminders to inform the stakeholders of their review assignments and deadlines.

Helix ALM subsequently permits annotation and commenting or annotate directly on the requirements. Discussions and cooperation between reviewers and the requirements author can be facilitated by comment threads. The tool also includes measurement and reporting features. Helix ALM has metrics and reporting capabilities that give its users insight into the review procedure. It can provide reports that highlight open issues, pinpoint delays, and summarize the review status. Project managers and other stakeholders can monitor progress and make wise decisions with the aid of these insights.

In conclusion, the analysis of the existing tools has revealed its strengths and weaknesses, offering valuable insights for the Collaborev project. In addition, the authors [13] also discussed the necessity of having tools to automate the process to detect requirements defects. By focusing on checklist-based technique, improving the user interface, implementing report generation functionality, and introducing a document tracking feature, Collaborev aims to enhance the review experience for all users. These enhancements will empower users to conduct more comprehensive reviews, streamline the documentation process, and facilitate effective collaboration.

III. METHODOLOGY

For this project, the prototyping process model was used as the project development methodology. As shown in Fig. 4, this approach involves continuous iteration of the prototype, allowing for incremental development and refinement. Through user feedback and testing at each iteration, stakeholders' feedback can be sought to ensure that Collaborev aligns closely with users' needs.

A. Requirements Engineering (RE) phase

At the beginning of the project, an extensive research and analysis were conducted. A literature review was carried out, and a comparison with other existing requirements review tools was performed. This provided valuable insights into the challenges faced during the requirements review process. By analyzing the gathered information, the project scope, features, and functionalities of the system, objectives, and improvements to offer were defined.

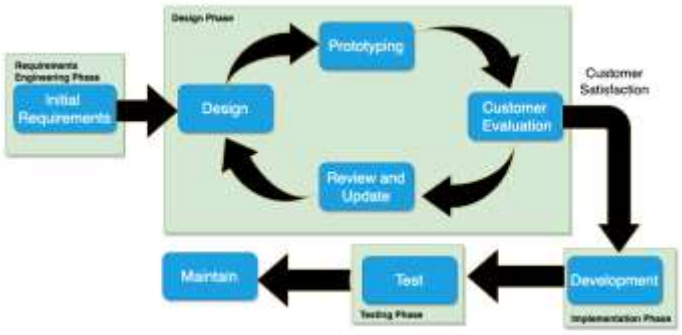

Fig.4 Prototyping Process Model

A requirement statement describes what behaviour a system should exhibit, without providing how they should

https://doi.org/10.31436/ijpcc.v10i1.456

be accomplished [1]. Hence, a simplified Software Requirements Specification (SRS) was prepared, focusing on the list of functional requirements (see Table I), and quality requirements.

In addition to the identified functional requirements, a set of quality requirements was also defined. Table II lists all the quality requirements that are relevant to the system, which includes security, performance and availability requirements.

TABLE I
COLLABOREV MODULES (FUNCTIONAL REQUIREMENTS)

No	Module name	No of functional requirements
1	Authentication module	5
2	Account module	2
3	Reviewer module	9
4	Document author module	6
5	Review leader module	7
6.	User and project management module	3
7.	Checklist management	3
	Total	32

Security requirements cover the authentication, authorization, and access features. While performance requirements focus on the acceptance criteria for uploading and downloading SRS documents to be reviewed, and finally availability requirement addresses on ensuring the system is available for the stakeholders to perform their tasks.

TABLE II
COLLABOREV MODULES (QUALITY REQUIREMENTS)

No	Module name	No of quality requirements
1	Security requirements	4
2	Performance requirements	4
3	Availability requirements	1
	Total	9

A use case diagram that depicts the actors and its interactions was created. Refer Fig. 5. In this diagram, four actors were defined i.e. reviewer, document author, review leader, and administrator. Reviewers are the main role in this project, who are responsible to provide the input to the requirements review for all the provided requirements for a particular project.

Document authors are those who provide the requirements to be reviewed. In addition, the review leader's role is to manage the review process by assigning the respective reviewers, while the administrator's role is to manage the use of the system from the perspectives of the users and projects.

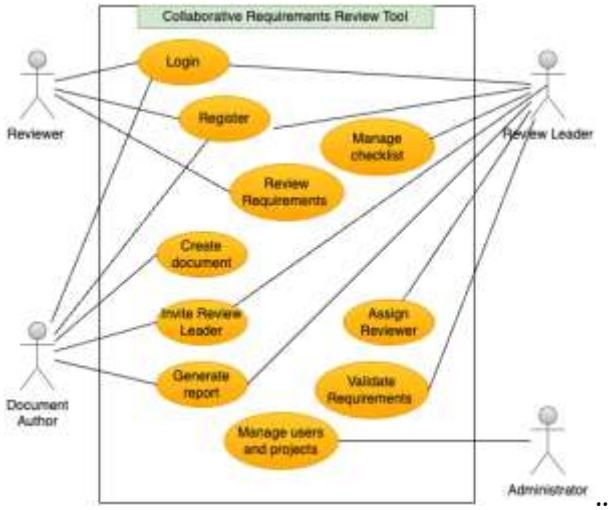

Fig. 5 Use case diagram for Collaborev

The generic process flow for Collaborev was adopted from e-Review process flow. In general, all the flows were replicated from eReview process flow except that in this project only checklist-based reading was included.

The review process begins when the document author provides the requirements to be reviewed. A review leader will be assigned, and he will then select the review team members. Each of the reviewer will access the system to perform the review. After the review is completed, the review leader will generate the consolidated review feedback. The report will then be sent to the document author.

B. Design Phase

During the design phase, the system architecture was created as the overview of the design process. Refer to B1 Architectural Design Subsection. The graphical user interface (GUI) design was created with Figma [14], providing the document authors, review leaders, administrators, and reviewers with straightforward interfaces. Refer to the B2 Subsection. Following that, a prototype was constructed based on the designed GUI. User feedback was used throughout the design process to improve and iterate GUI designs and a prototype.

In the design phase, a systematic approach was followed to create an efficient system. The collected requirements were analyzed, and algorithms were designed to perform specific tasks, ensuring accuracy and efficiency. The design of the system was focused on defining the overall architecture and components. Refer Sub-section B1 Architectural Design. The graphical user interfaces (GUI) was designed using Figma [14], ensuring it was intuitive and easy to navigate for the users, including the author, review leader, administrators, and reviewers. Refer Sub-section B2. A prototype was developed to replicate the GUI

https://doi.org/10.31436/ijpcc.v10i1.456

that has been designed. Throughout the design phase, the GUI designs and the prototype were iterated and refined based on users' feedback.

1) *Architectural Design:* The Model-View- Controller (MVC) architecture is a common and widely adopted software design pattern used to build maintainable and scalable applications [15]. The adoption of the MVC framework played a crucial role in shaping the design to facilitate efficient data storage and retrieval. The emphasis was placed on establishing a seamless interaction between various system modules, ensuring a coherent and organized development approach that aligns with the MVC best practices.

The MVC fundamental features of application components, such as database access and transaction management, are included in the *model* component. It captures the system current state and applies the respective transformations to it. Usually, neither the view nor the controller is particularly known to the model. The state that the model represents is shown by the *view*. It controls how the system are displayed visually. In particular, the view does not contain any processing logic; its only role is to fetch items from the model. When the model's state changes, the view ought to be informed. In addition, it has no knowledge of the controller. Finally, the user interaction with the model is managed by the *controller*. It oversees the processing of requests and the production of any elements that the view needs. Furthermore, based on the user needs, it transmits the user request to the view [16].

One of the primary results of the project was the meticulous integration of the MVC framework into Collaborev's system architecture as depicted in Fig. 6. In this phase, the system's architecture was defined, including its overall structure, components, and the relational database model.

When there is a request from user, it routes the request to a controller, which interacts with the data model. This model returns the result back to the controller layer and the controller invokes the view layer. The view layer renders the result to be displayed to the user

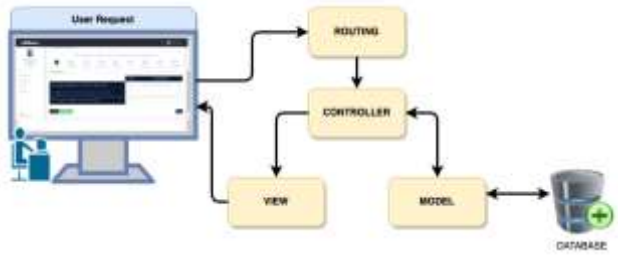

Fig. 6 Collaborev Architecture Diagram

2) *Graphical User Interface (GUI) Design:* The reviewer dashboard, where all the review related details as well as the document author's view when requirements are uploaded to the tool were set out. This follows with the review main content where the reviewers go through each requirement to look for any requirements errors. Subsequently, Finally, the report generation feature after the review process is completed.

3) *Data/Class Design:* To support the storage and management of data, the database model was designed, which later was used to define the tables, relationships, and attributes required for effective data handling. In Fig. 7, all the classes were designed based on the MVC architecture and all the interface(boundary), controller, and entity classes were included.

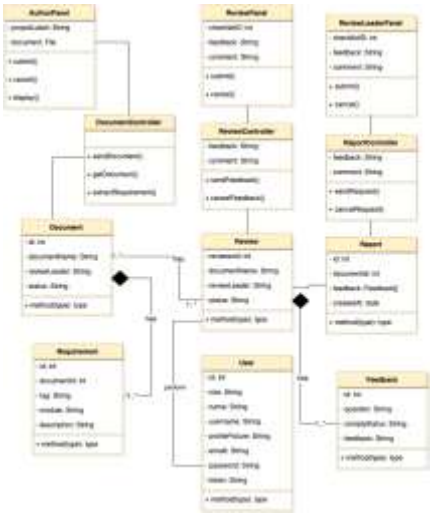

Fig. 7: Class diagram for Collaborev

C. *Implementation Phase*

The project proceeded to the implementation phase following the completion of the design phase. The database was built using MySQL, where tables were created, relationships were defined, and the structure was optimized for efficient data storage and retrieval. The user interface was developed, translating the designs into implementation using PHP, HTML, Bootstrap and CSS.

The author dashboard is where the main information is displayed to the author such as information of the assigned review tasks, the status of the review whether it is in progress of completed, or not reviewed yet. The reviewer dashboard is the next where the requirements are displayed to the reviewer such as information of the assigned review tasks, the status of the review whether it is in progress of completed, or not reviewed yet. The system detects each of the requirements to be reviewed in the review.

https://doi.org/10.31436/ijpcc.v10i1.456

The subsequent feature dwells on the feedback report generated after the review session ends. This report integrates all the review feedback from all the assigned reviewers to be used by the document owner

Fig. 8 Review leader report

Fig. 8 demonstrates the status of the review from the review leader's perspective. This report will be sent to the document author so that he would be able update the respective SRS document based on the review feedback.

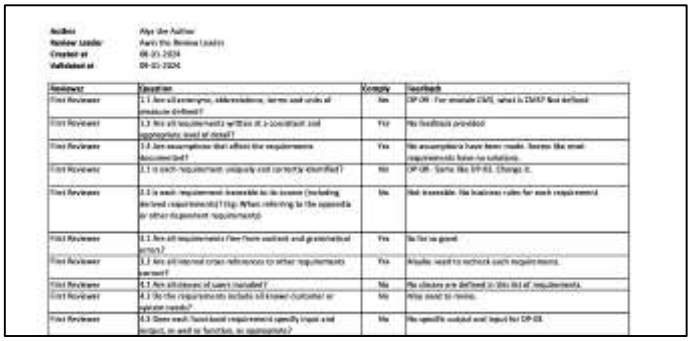

Fig. 9 Generate detailed review report

Finally, a sample of the detailed outcome of the requirements review process is shown in Fig. 9. The review leader shall be able to generate the detailed review report in pdf or Excel format.

C. *Testing Phase*

The testing phase was crucial to ensuring the quality and reliability of the system. Test cases were drafted for functional testing to validate that the system satisfies the requirements, and system specifications. This involved creating test cases by specifying inputs, expected outcomes, and any specific conditions or constraints. Nonetheless, the testing effort was minimal during the project execution due to some technical issues during deployment process, and time constraint.

Table III lists a sample of test cases created for the project. During testing, these test cases were executed to ensure all the requirements were satisfied. If there is any error, those errors were fixed, and the relevant features were updated. This is to ensure that in the end all requirements have been tested.

TABLE III
TEST CASES SAMPLE

Test Case ID	Feature Name	Tester Criteria	Summary	Precondition	Execution Steps	Expected Result
AUT-01	User Registration	NA	To verify that user can register their account.	N/A	1. Navigate to the registration page. 2. Enter valid personal details. 3. Submit the registration form.	A new account is created, and the user is redirected to the login page.
AUT-02	User Login	NA	To verify that user can login into their account.	User has registered an account.	1. Navigate to the login page. 2. Enter a valid email address and password. 3. Click the login button.	The user is successfully logged into the system.
AUT-03	Invalid Login	NA	To verify that user cannot login into their account with invalid credentials.	N/A	1. Navigate to the login page. 2. Enter an invalid email address or password. 3. Click the login button.	An error message is displayed, indicating invalid credentials.
AUT-04	Logout	NA	To verify that user can logout off their account.	User has logged in into their account.	1. Click on the logout button.	The user is logged out, and the system redirects to the login page.
AUT-05	Edit Personal Details	NA	To verify that user can edit their personal details.	User has logged in into their account.	1. Navigate to the user profile editing page. 2. Modify personal details as per Business Rule BR-03. 3. Save the changes.	Personal details are successfully updated.
AUT-06	Account Deletion	NA	To verify that user can delete their account.	User has logged in into their account.	1. Navigate to the account deletion page. 2. Confirm the account deletion.	The account is deleted, and the user is redirected to the registration page.

IV. Result And Discussion

Overall, proper documentation was maintained, and effective collaboration within the team was ensured throughout the project execution. Feedback and user testing from professionals or experts were incorporated to iteratively refine designs and implementation, ensuring that the system met project goals and objectives.

Based on the outcomes of the project, the scope and objectives were accomplished. The project's main aim is to enhance the existing e-Review system [9] by incorporating additional features and functionalities. These include collaborative and simplified review capabilities, streamlined feedback consolidation, and automated review report generation. The target users of Collaborev are requirements reviewers, requirements review leaders, and authors of the requirements.

V. Conclusions

In summary, Collaborev's development marks an important turning point in the field of software engineering, particularly for the requirements review procedure. The thorough design and architecture, together with the thoughtful integration of the MVC architecture and Laravel framework, highlight our dedication to building a reliable, effective, and user-friendly web-based system. Collaborev is tightly aligned with the changing needs of its stakeholders because to the iterative prototyping technique, which has enabled continual refinement based on insightful user input. By addressing the key issues with traditional requirements

review procedures, this project provided a solution that improved cooperation, increased overall software development lifecycle efficiency and accuracy, and streamlined communication. Collaborev demonstrates the effectiveness of implementing contemporary frameworks and techniques, paving the way for further advancement and development in software engineering processes going forward.

Future developments may concentrate on a comprehensive assessment, ongoing modifications, and improvements contingent upon practical implementation and evolving industry norms. It will be crucial to identify areas for enhancement and new features based on feedback and observations from users to increase its value. To make sure the system achieves the project's goals and objectives, however, professional or RE specialists' feedback and user testing will need to be added in the future.

In conclusion, Collaborev creates the foundation for a ground-breaking tool in RE. Future efforts will focus on enhancing the system's real-world implementation, exploring state-of-the-art technologies for improvement, and tailoring the system to meet the intricate needs of diverse software development fields.

ACKNOWLEDGMENT

The authors would like to extend our appreciation to the Kulliyyah of Information and Communication Technology (KICT), International Islamic University Malaysia (IIUM) as well as the Computer Science Department for the opportunity to work for this project.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest

REFERENCES

[1] N. Carlson and P. Laplante, 'The NASA automated requirements measurement tool: a reconstruction', *Innov. Syst. Softw. Eng.*, vol. 10, no. 2, pp. 77–91, Jun. 2014, doi: 10.1007/s11334-013-0225-8.

[2] M. A. Akbar, Nasrullah, M. Shameem, J. Ahmad, A. Maqbool, and K. Abbas, 'Investigation of Project Administration related challenging factors of Requirements Change Management in global software development: A systematic literature review', in *2018 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube)*, Quetta: IEEE, Nov. 2018, pp. 1–7. doi: 10.1109/ICECUBE.2018.8610966.

[3] Z. Liu, B. Li, J. Wang, and R. Yang, 'Requirements engineering for crossover services: Issues, challenges and research directions', *IET Softw.*, vol. 15, no. 1, pp. 107–125, 2021, doi: 10.1049/sfw2.12014.

[4] G. Kotonya and I. Sommervile, *Requirements engineering: Processes and techniques*. West Essex, England: John Wiley, 2003.

[5] A. Nordin, N. Z. Abidin, and S. H. M. Zaini, 'Collaborative requirements review', *Int. J. Eng. Technol.*, vol. 7, no. 2.14, Art. no. 2.14, Apr. 2018, doi: 10.14419/ijet.v7i2.14.11158.

[6] E. Bjarnason *et al.*, 'Challenges and practices in aligning requirements with verification and validation: a case study of six companies', *Empir. Softw. Eng.*, vol. 19, no. 6, pp. 1809–1855, Dec. 2014, doi: 10.1007/s10664-013-9263-y.

[7] A. Nordin, M. M. Islah, and R. Z. Abdul, 'Design and Development of the e-Review: An Online Requirements Review Tool based on Reading Techniques', *Int. J. Perceptive Cogn. Comput.*, vol. 9, no. 1, Art. no. 1, Jan. 2023, doi: 10.31436/ijpcc.v9i1.375.

[8] A. Nordin, N. Z. Abidin, and S. H. M. Zaini, 'Collaborative requirements review', *Int. J. Eng. Technol.*, vol. 7, no. 2, pp. 66–69, 2018, doi: 10.14419/ijet.v7i2.14.11158.

[9] I. M. Musleh, A. Nordin, and N. A. Emran, 'An Experimental Study on Checklist-Based and Perspective-Based Requirements Reading Techniques Using E-Review Tool', *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 28, no. 3, pp. 351–367, 2022, doi: 10.37934/araset.28.3.351367.

[10] I. M. Musleh, A. Nordin, and N. A. Emran, 'An Experimental Study on Checklist-Based and Perspective-Based Requirements Reading Techniques Using E-Review Tool', *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 28, no. 3, Art. no. 3, Nov. 2022, doi: 10.37934/araset.28.3.351367.

[11] 'Requirements Management Software'. Accessed: Jan. 13, 2024. [Online]. Available: https://specinnovations.com/innoslate/requirements-management-software

[12] 'Helix ALM | ALM Tool | Perforce'. Accessed: Jan. 15, 2024. [Online]. Available: https://www.perforce.com/products/helix-alm

[13] M. Kamalrudin, L. L. Ow, and S. Sidek, 'Requirements Defects Techniques in Requirements Analysis: A Review', *J. Telecommun. Electron. Comput. Eng. JTEC*, vol. 10, no. 1–7, Art. no. 1–7, Feb. 2018.

[14] 'Figma: The Collaborative Interface Design Tool', Figma. Accessed: Jan. 16, 2024. [Online]. Available: https://www.figma.com/

[15] Z. Tu, 'Research on the Application of Layered Architecture in Computer Software Development', *J. Comput. Electron. Inf. Manag.*, vol. 11, no. 3, pp. 34–38, 2023.

[16] Y. Ping, K. Kontogiannis, and T. C. Lau, 'Transforming legacy Web applications to the MVC architecture', in *Eleventh Annual International Workshop on Software Technology and Engineering Practice*, Sep. 2003, pp. 133–142. doi: 10.1109/STEP.2003.35.