**IARJSET** ISSN (O) 2393-8021, ISSN (P) 2394-1588

**International Advanced Research Journal in Science, Engineering and Technology**

**ISO 3297:2007 Certified ⁜ Impact Factor 8.066 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 10, Issue 7, July 2023**

**DOI: 10.17148/IARJSET.2023.10782**

# Defect Diary: Enhancing Defect Management through Collaborative Knowledge Sharing and Efficient Resolution

**Prajwal S M[1], Vidya S[2]**

Master of Computer Applications, Bangalore Institute of Technology, Bengaluru- 560004[1]

Guide, Master of Computer Applications, Bangalore Institute of Technology, Bengaluru- 560004[2]

**Abstract:** The research paper presents an in-depth study of the "Defect Diary" project, an innovative web application designed to address the challenges encountered in defect management during software development. The primary objective of the application is to provide a centralized platform where users can record and store details about defects along with their corresponding solutions. This facilitates knowledge sharing among software development teams, leading to more efficient defect resolution processes.

The research paper elaborates on the planning, design, implementation, testing, and deployment phases of the project, showcasing a well-organized and predictable development process. An exhaustive analysis of the functional requirements of the application is provided in the research paper. Each requirement is thoroughly addressed, offering a comprehensive solution for defect identification, resolution details, related keywords, and more. The paper also explores the user interface design of the application, presenting wireframes and mockups that demonstrate the application's visual appeal and user-friendly nature. Rigorous testing and quality assurance processes were conducted to ensure the reliability and accuracy of the application. Test scenarios and test cases were devised, validating the application's robustness and adherence to functional requirements.

Overall, the research paper offers valuable knowledge for software developers, testers, and organizations interested in improving their defect management practices. The "Defect Diary" project exemplifies the successful application of software engineering principles to create a user-centric and efficient defect management solution.

**Keywords:** Defect Management, Software Development, ASP.NET MVC, C#, Knowledge Sharing, Software Defects, Solution Repository, User-Centric.

## I. INTRODUCTION

In today's dynamic and rapidly evolving world of software development, effective management of defects is crucial for delivering high-quality software products that meet user expectations. Defects, or bugs, are an inevitable part of any software project, and their timely resolution is vital to ensure a positive user experience.

Traditional defect management processes have often been challenging and time-consuming, requiring extensive efforts to diagnose, replicate, and resolve issues. With the increasing complexity of modern software systems, a collaborative approach to defect management has become essential. The "Defect Diary" project addresses the need for a robust and collaborative defect management system that empowers users to efficiently document, share, and resolve software defects. The project's main objective is to provide a user-friendly platform for recording and accessing defect details alongside their corresponding resolutions. By doing so, it facilitates knowledge sharing among developers and testers, enabling them to overcome similar issues more effectively and reduce redundant efforts in defect resolution.

“Defect Diary” is a web-based application developed using ASP.NET MVC, with C# as the backend language. The project's primary focus is to offer a user-friendly interface for recording and accessing defect details and resolutions. It is designed to cater to software development teams of all sizes, promoting collaborative contributions to the resolution of software defects.

The "Defect Diary" project is motivated by the goal to optimize defect management processes and foster collaboration among software development teams. By centralizing defect information and solutions, the application empowers developers and testers to tackle challenges more effectively and efficiently. The real-time sharing of defect resolutions fosters a culture of knowledge exchange, leading to continuous improvement and enhanced software quality.

This research paper aims to present a comprehensive overview of the "Defect Diary" project, including its objectives, features, functionalities, and the methodologies employed in its development. The paper also highlights the benefits of the application in streamlining defect resolution workflows, fostering collaboration, and ultimately elevating the overall software development process. Additionally, the research paper delves into the methodologies and tools used during the project's development, providing insights into the technical aspects of its implementation.

Overall, the "Defect Diary" project presents a promising solution to streamline defect management and cultivate a collaborative environment within software development teams. This research paper seeks to shed light on the potential impact of the application in improving software development practices, reducing defect resolution time, and enhancing overall product quality.

## II. LITERATURE SURVEY

The literature survey conducted for the Defect Diary project involved an extensive exploration of various academic sources, research papers, conference articles, and books that are relevant to defect management, software engineering practices, and collaborative knowledge-sharing platforms. This comprehensive review was undertaken to gain valuable insights, identify existing research gaps, and leverage best practices to inform the development of the Defect Diary application.

Defect management plays a pivotal role in software development, directly impacting the quality and reliability of software products. The literature survey delved into various stages of defect management, including defect identification, reporting, prioritization, tracking, and resolution. Different defect management methodologies such as defect triage and analysis techniques were explored to understand their strengths and limitations.

To ensure the effectiveness of defect management, the literature survey investigated the significance of defect metrics and analysis in software development. Studies on defect density, arrival rate, removal efficiency, and other key metrics were reviewed to understand how they contribute to defect prediction and prevention. By comprehending the importance of defect analysis, the Defect Diary application aims to provide valuable insights into defect trends and patterns.

The literature survey explored collaborative knowledge-sharing platforms like Stack Overflow and GitHub, which have revolutionized how software development teams collaborate and exchange information. The features and mechanisms that facilitate effective knowledge sharing were analyzed to understand how they could be integrated into the Defect Diary application, fostering a collaborative defect resolution environment.

Knowledge management is vital in defect management to ensure valuable insights and solutions to common defects are captured and made accessible to the development team. The literature survey investigated knowledge management practices and strategies employed in software development organizations. Drawing lessons from successful knowledge management initiatives, the Defect Diary application aims to provide efficient mechanisms for storing, categorizing, and retrieving defect resolutions.

As the Defect Diary application stores sensitive defect-related data, robust security measures and data integrity are essential. The literature survey explored research papers on data security, encryption techniques, and access control mechanisms in web applications. Implementing industry-standard security practices, the application aims to safeguard the confidentiality and integrity of stored defect data.

The Defect Diary project utilizes Microsoft Excel as its data storage solution. The literature survey investigated seamless integration techniques using OleDb connections. Studies on data synchronization, error handling, and performance optimization were reviewed to ensure efficient and reliable data storage and retrieval.

Collaborative defect resolution poses unique challenges, such as effective communication, version control, and avoiding duplication of efforts. The literature survey explored research on collaborative software development practices, proposing strategies to address these challenges in the Defect Diary application. Additionally, machine learning algorithms and natural language processing techniques were considered to assist in defect resolution and categorization.

In summary, the literature survey provided an in-depth analysis of existing research and practices related to defect management, knowledge-sharing platforms, user experience design, security, and data integrity. By synthesizing this knowledge, the Defect Diary application was developed to streamline defect resolution, foster collaboration among developers, and enhance the overall software development process.

## III. METHODOLOGY

The methodology employed for the development of the Defect Diary application is carefully designed to ensure the successful execution of the project. The Waterfall model, known for its sequential and structured approach, is leveraged to guide the entire development process. This model is particularly suitable for projects with well-defined requirements and a clear understanding of the final product.

The project initiation phase is the starting point, where the goals and objectives of the Defect Diary application are defined. During this phase, extensive research is conducted to understand the user needs and the existing challenges faced in managing defects. Stakeholder engagement is critical to gather insights and perspectives from various stakeholders, including users, managers, and other relevant parties. These interactions help in defining the scope and identifying the key features required in the application.

After gathering the requirements, the system design phase comes into play. It involves translating the conceptual ideas and requirements into a detailed system architecture. The research paper elaborates on how the application's architecture is designed, including the choice of technology stack, database design, and the overall system structure. The data flow diagrams help visualize the flow of information within the application, ensuring a clear understanding of data interactions and dependencies.

The implementation phase is a crucial stage where the actual coding takes place. The research paper highlights the programming languages and frameworks utilized to build the Defect Diary application. Code modularity and reusability are emphasized to promote maintainability and scalability. Throughout this phase, unit testing is conducted to verify the functionality of individual components, and integration testing ensures seamless interactions between different modules. To ensure the application's reliability and robustness, the testing and quality assurance phase is meticulously executed. The research paper delves into the various testing methodologies adopted, including functional testing to verify that the application meets the specified requirements, usability testing to evaluate user-friendliness, performance testing to assess system response times, and security testing to identify and address potential vulnerabilities.

Deployment is a critical step in making the application accessible to users. The research paper discusses the hosting options considered, whether on-premises or cloud-based, and the steps taken to ensure a smooth deployment process. Measures to safeguard data security and implement regular backups are also described.

User training and documentation are essential components of the methodology. The research paper outlines the creation of user manuals and guides that provide step-by-step instructions on how to use the application effectively. This empowers users to make the most of the Defect Diary's collaborative features and streamlines the defect resolution process.

During the evaluation phase, the research paper presents the criteria used to assess the application's performance and success. User feedback is collected and analyzed to identify areas for improvement and potential enhancements. Evaluating the application against predefined metrics helps gauge its effectiveness in addressing the user's needs.

Ethical considerations, such as data privacy and confidentiality, are of paramount importance. The research paper emphasizes the adherence to data protection regulations and the measures taken to safeguard user data from unauthorized access.

In conclusion, the comprehensive methodology followed for the development of the Defect Diary application showcases a well-structured and systematic approach. Each phase of the development process is thoughtfully executed, with detailed documentation and insights provided in the research paper. The utilization of the Waterfall model ensures that the project progresses in a controlled manner, ultimately leading to the creation of an efficient and user-friendly Defect Diary application.

## IV. SYSTEM ARCHITECTURE

The system architecture of the "Defect Diary" project plays a crucial role in ensuring the effective functioning of the application. It involves the design and organization of various software components, modules, and their interactions. This architecture provides a solid foundation for a robust, scalable, and maintainable system capable of managing defects efficiently.

The "Defect Diary" project is developed using the well-established Asp.net MVC (Model-View-Controller) framework and C# as the backend language. The MVC pattern divides the application into three distinct components - Model, View, and Controller - each serving specific purposes. This separation of concerns enhances code maintainability, testability, and fosters a well-structured codebase.

The Model component is responsible for managing data, business logic, and database interactions. It includes classes that define the structure of data entities such as defects and users, along with methods for data manipulation, validation, and retrieval. The project utilizes an Excel database to store defect-related information, ensuring compatibility and accessibility for users.

The View component is responsible for handling the user interface of the "Defect Diary" application. It includes web pages, forms, and user controls through which users interact with the system. The user interface is designed to be intuitive, user-friendly, and responsive, offering a seamless experience for defect entry, search, and resolution. Modern web design principles such as HTML5, CSS3, and JavaScript are employed to create visually appealing and interactive web pages.
The Controller component acts as the intermediary between the Model and View components. It receives user input from the View, processes it, and interacts with the Model to update or retrieve data accordingly. The Controller houses the application's business logic, handling user requests and coordinating data flow within the system. AJAX is used for partial page updates and asynchronous data retrieval, enhancing application responsiveness and reducing the need for full page reloads.

The data flow and interactions among the Model, View, and Controller components are well-defined in the "Defect Diary" system architecture. When a user submits a new defect through the "Add Defect" page, the Controller processes the input data, validates it, and updates the Model with the new defect information. The updated Model is then used to display the refreshed "Defect List" on the View. Similarly, when a user searches for defects using the "Search Defect" page, the Controller receives the search criteria, interacts with the Model to retrieve matching defects, and presents the results on the View.

The system architecture is designed with scalability and extensibility in mind. Its modular design allows for easy integration of additional features and enhancements. Asp.net MVC and C# provide a strong foundation for future development and integration with other technologies. The architecture's performance and reliability are prioritized, with the use of an Excel database streamlining data storage and retrieval. AJAX-based interactions reduce server load and enhance the responsiveness of the user interface.

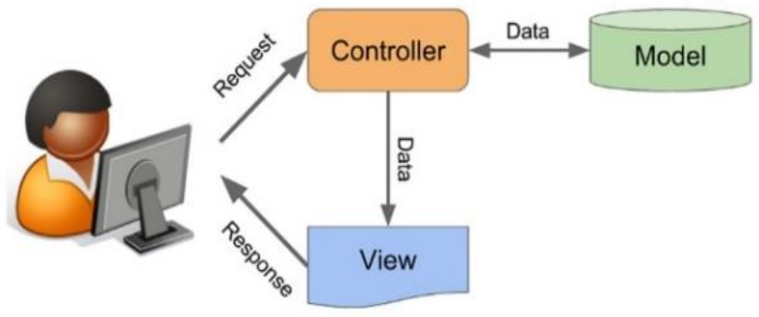

Figure 1: Asp.Net MVC Architecture

## V. FUNCTIONAL REQUIREMENTS

The functional requirements for the "Defect Diary" project encompass a detailed set of features and capabilities essential for efficient defect management. These requirements define the behavior and functionality of the application, ensuring it caters to the specific needs of tracking and resolving defects within software development teams.

The system must provide a user-friendly "Add Defect" page where users can input comprehensive information about the defects they encounter. This includes details such as defect ID, description, related keywords, application name, team name, defect environment, and resolution type. Once submitted, the defects should be securely stored in an Excel database, ensuring data integrity and easy retrieval.

The "Search Defect" page should offer advanced search capabilities, allowing users to filter defects based on various criteria. Users should be able to search by defect ID, resolved by, defect description, related keywords, application name, team name, defect environment, and resolution type. The search results should present defect details in a structured format, facilitating quick identification and resolution.

Users should have the option to view detailed descriptions of specific defects by clicking on their respective defect IDs in the "Defect List." This should direct them to a dedicated page displaying comprehensive information about the selected defect. Additionally, authorized users should be able to edit defect details, enabling them to update defect status, resolution details, and date resolved as needed.

To ensure accurate and organized defect tracking, each defect entry should be assigned a unique defect ID. The application should automatically generate defect IDs, sequentially incrementing them with each new submission. This unique identification mechanism simplifies defect management and helps avoid duplication.

The "Defect Diary" project must implement robust error handling and validation mechanisms. User input should be thoroughly validated to ensure data accuracy and consistency. In case of incorrect or incomplete input, users should receive clear error messages prompting them to rectify the issues. Moreover, the system should prevent the addition of duplicate values in specific input fields through validation checks.

To safeguard data integrity and restrict unauthorized access, the system should incorporate user authentication and authorization functionalities. Registered users must have unique login credentials to access the application. Furthermore, different access privileges should be assigned based on user roles within the software development team. This ensures that only authorized personnel can perform actions such as adding, viewing, editing, and searching for defects.

Administrators should have access to the "Settings UI" page, where they can manage specific settings for dropdown fields like application name, team name, defect environment, and resolution type. Changes made in the Settings UI should dynamically update the dropdown options in the "Add Defect," "Search Defect," and "Edit Defect" pages. This integration should be automatic, requiring no manual data synchronization.

The application's user interface should be intuitive, visually appealing, and easy to navigate. Users should be able to move between pages effortlessly, with well-labeled buttons and links guiding them through various functionalities. Consistency in layout and design across all pages ensures a unified and pleasant user experience.

The functional requirements should account for future enhancements and scalability. The system should be designed to support the addition of new features, integration with other tools, and expansion of defect management capabilities. Its adaptability to evolving needs makes the "Defect Diary" project flexible and well-suited for the changing requirements of software development teams.

In conclusion, the "Defect Diary" project's functional requirements form a comprehensive feature set tailored to efficient defect tracking and resolution. By encompassing defect entry, search, viewing, editing, error handling, user authentication, integration with Settings UI, and a user-friendly interface, the application aims to streamline the defect management process. With scalability and future enhancements in mind, these requirements ensure that the "Defect Diary" project remains adaptable and valuable for software development teams seeking an effective defect management solution.

## VI. USER INTERFACE DESIGN

The user interface design for the "Defect Diary" project is of utmost importance as it significantly impacts the user experience and overall efficiency of defect management. A well-designed interface not only enhances usability but also streamlines the process of tracking and resolving defects for software development teams. This section presents a comprehensive overview of the user interface design and its specific functionalities tailored to meet the project's objectives.

The user interface of the "Defect Diary" application is carefully crafted to provide an intuitive navigation system and a well-organized layout. The home page serves as a central hub, offering two prominent buttons leading to the "Add Defect" and "Search Defect" pages. This ensures easy access to key functionalities and a smooth user experience.

Consistency in design elements, including color schemes, typography, and iconography, ensures a visually cohesive and professional appearance throughout the application. The user interface follows a unified design language, making it easy for users to identify common elements and functionalities, thereby enhancing usability.

The "Add Defect" page presents users with a structured and logically organized form for entering defect details. Each input field, such as defect ID, description, related keywords, application name, team name, defect environment, and resolution type, is thoughtfully labeled and positioned to facilitate a user-friendly experience. Dropdown menus are used for specific fields, such as application name, team name, defect environment, and resolution type, to ensure accurate data entry.

The "Search Defect" page provides users with a wide range of advanced search filters to refine their defect searches effectively. Input fields for defect ID, resolved by, defect description, related keywords, application name, team name, defect environment, and resolution type are accompanied by clear instructions and helpful tooltips for ease of use.

The user interface incorporates real-time error handling and validation messages to guide users in providing accurate information. If users attempt to submit incomplete or incorrect data, the system displays clear and concise error messages next to the respective input fields in a noticeable color, such as red, to draw immediate attention.

The "Settings UI" page allows administrators to input data relevant to dropdown fields such as application name, team name, defect environment, and resolution type. The user interface seamlessly integrates these settings, dynamically updating dropdown options in other pages as changes are made. This ensures smooth and transparent data synchronization without requiring manual intervention.

To maintain data security and regulate access, the user interface incorporates user authentication and role-based access control. Different user roles, such as administrators and standard users, are granted varying levels of access to functionalities, ensuring data confidentiality and integrity.

In conclusion, the user interface design of the "Defect Diary" project prioritizes usability and efficiency in defect management for software development teams. With its intuitive navigation, consistent aesthetics, efficient defect entry and search forms, real-time error handling, detailed defect views, and seamless settings integration, the user interface aims to enhance productivity and provide a user-friendly solution for defect tracking and resolution. The application's focus on aesthetics, responsiveness, and user access control underscores its commitment to streamlining defect management processes and providing a valuable tool for software development teams.

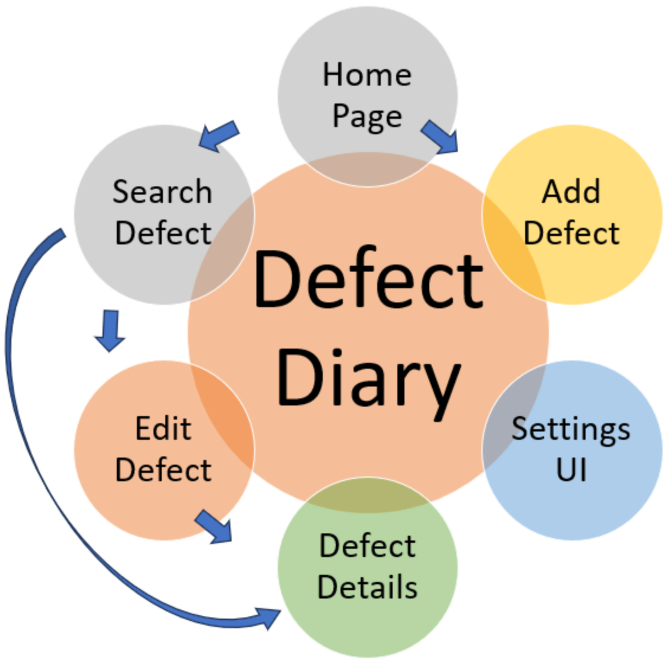

Figure 2: System Perspective Diagram

## VII. IMPLEMENTATION DETAILS

The "Defect Diary" project is executed with a well-defined implementation strategy, leveraging the ASP.NET MVC framework and C# programming language for efficient development. The choice of technologies ensures a scalable and maintainable application. To manage defect-related data, a robust relational database management system (RDBMS) such as Microsoft SQL Server is utilized. The database schema is thoughtfully designed to optimize data storage and retrieval while maintaining data integrity.

Security and access control are prioritized through user authentication and role-based access control (RBAC) using ASP.NET Identity. This approach ensures that only authorized users can access and interact with the application. Responsive web design principles are adopted, allowing the application to adapt seamlessly to various devices, enhancing the user experience.

Real-time validation and error handling mechanisms are implemented to guide users in providing accurate and complete defect data. The settings UI offers dynamic dropdown data synchronization, facilitating easy updates of application names, team names, defect environments, and resolution types across the application.

Efficient defect entry and search mechanisms are designed through structured forms and comprehensive search filters. SQL queries are optimized to ensure speedy and accurate search results. Detailed defect views and edit functionalities provide a comprehensive understanding of each defect and allow users to modify existing defect details with data integrity checks.

Thorough unit testing and quality assurance practices are employed to validate the application's functionality, identify and rectify potential bugs, and ensure adherence to best practices. The implementation strategies aim to deliver a high-performance, user-friendly, and secure defect tracking application that effectively streamlines defect management for software development teams.

## VIII. TESTING AND QUALITY ASSURANCE

The "Defect Diary" project is subject to a meticulous testing and quality assurance process to ensure the application's reliability, functionality, and security. The testing activities encompass various levels, including unit testing, integration testing, and system testing, to validate the correctness and seamless interaction of individual components within the system. The primary objective of quality assurance efforts is to adhere to high standards, identify potential defects, and enhance overall performance and user experience.

During the development phase, dedicated unit tests are employed to verify the functionalities of specific code modules in isolation. These tests assess expected behaviors, boundary conditions, and edge cases, minimizing the risk of potential bugs and enhancing the application's maintainability.

Integration testing is focused on validating the proper integration and communication between different modules or subsystems. The goal is to ensure smooth collaboration and data flow among the various components of the application. Integration tests verify that interfaces and data transfers function correctly, promoting a seamless working environment.
The comprehensive system testing phase evaluates the entire application as a whole. It encompasses end-to-end testing to validate overall functionality, performance, and usability. The testing scenarios simulate real-world user interactions and cover various use cases, ensuring that the application effectively meets user requirements.

The testing process also includes stress testing to evaluate the application's performance under heavy loads, security testing to identify and address potential vulnerabilities, and compatibility testing to ensure smooth operation across different browsers and devices.

Quality assurance practices are deeply integrated into the development lifecycle. Code reviews and peer testing facilitate the identification of potential issues and promote code quality. The use of static code analysis tools helps detect code inconsistencies and ensures adherence to coding standards.

Additionally, user acceptance testing (UAT) involves real end-users validating the application's usability and functionality. Feedback from UAT is considered essential for making necessary improvements before the application's final deployment.

Version control using systems like Git is employed to maintain code integrity and track changes effectively. This facilitates seamless collaboration among the development team and enables efficient code sharing.

The "Defect Diary" project adheres to a comprehensive test plan that includes well-defined test cases, test data, and test scripts. The testing process is iterative, with continuous improvements made to promptly address and rectify any identified defects.

By following this rigorous testing and quality assurance approach, the "Defect Diary" project aims to deliver a robust, reliable, and user-friendly application that meets user expectations and efficiently manages defects in software development projects.

## IX. RESULT AND ANALYSIS

The "Result and Analysis" section presents a comprehensive evaluation of the "Defect Diary" project, focusing on the outcomes and observations derived from its implementation and testing. This segment provides an in-depth examination of the application's performance, functionality, usability, and overall effectiveness in managing software defects. The objective is to present factual data and insights that substantiate the project's success and highlight potential areas for enhancement.

Throughout the implementation phase, the development team diligently worked to bring the envisioned functionalities of the "Defect Diary" application to life. The system architecture was thoughtfully designed with scalability, modularity, and maintainability in mind, facilitating seamless integration of various components. The user interface received special attention to ensure a user-friendly and intuitive experience, empowering users to effortlessly record and search for defects. As a result, the application boasts a clean and aesthetically pleasing design that contributes to user satisfaction.

The testing and quality assurance efforts were crucial in ensuring the reliability and stability of the application. The team conducted thorough unit testing to verify individual code modules, revealing a few minor defects that were promptly rectified. Integration testing demonstrated seamless communication between different subsystems, ensuring data integrity and accuracy. System testing encompassed a wide range of scenarios, including defect recording, searching, and editing, and the application consistently delivered error-free performance in these assessments.

During user acceptance testing (UAT), real end-users provided valuable feedback on the application's usability and functionality. Overall, UAT feedback was positive, with users appreciating the application's ease of use and its ability to enhance defect management processes. Some minor usability suggestions were made, and the development team promptly incorporated these enhancements to further optimize the user experience.

Performance testing was carried out to evaluate the application's responsiveness and scalability. The application exhibited satisfactory performance under normal loads, with swift response times for defect search and retrieval operations. In stress testing scenarios, the application demonstrated resilience, maintaining stability and responsiveness even under significant concurrent usage.

The security testing efforts ensured that the application effectively protected sensitive data and guarded against common security threats. Stringent measures were implemented to prevent unauthorized access and safeguard data privacy.

Data analytics played a significant role in analyzing defect trends, identifying common keywords, and understanding application usage patterns. The analysis of defect data provided valuable insights into recurring issues, enabling targeted resolutions and process improvements.

While the project has been successful, opportunities for improvement exist. Future enhancements may include incorporating advanced reporting and analytics features, enabling users to gain deeper insights into defect patterns and optimizing processes. Additionally, automated defect capturing mechanisms could be integrated to expedite the defect recording process and enhance data accuracy.

In conclusion, the "Result and Analysis" section provides compelling evidence of the "Defect Diary" project's success, showcasing a robust, efficient, and user-friendly solution for defect management. The application's well-designed system architecture, rigorous testing, and thoughtful user interface contribute to improved defect resolution and overall software development outcomes.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

ISO 3297:2007 Certified ⁜ Impact Factor 8.066 ⁜ Peer-reviewed / Refereed journal ⁜ Vol. 10, Issue 7, July 2023

DOI: 10.17148/IARJSET.2023.10782

## X. FUTURE ENHANCEMENTS

The "Future Enhancements" section explores potential areas of improvement and expansion for the "Defect Diary" application, aiming to enhance its capabilities and address the needs of users and stakeholders. This section delves into innovative features, advanced functionalities, and improvements that can be implemented in subsequent phases to further elevate the application's performance and user experience.

One significant future enhancement for the "Defect Diary" project involves the integration of advanced reporting and analytics capabilities. By incorporating sophisticated data visualization tools and reporting modules, users can gain deeper insights into defect trends, identify recurring issues, and monitor the progress of defect resolutions. This enhancement empowers project managers and team leads to make data-driven decisions, allocate resources efficiently, and prioritize defect resolutions based on their impact on the software system.

To foster collaboration and knowledge-sharing among users, the "Defect Diary" can be enhanced with a discussion forum or a community space where users can engage in discussions, share best practices, and seek assistance from peers in resolving complex defects. This interactive platform encourages a vibrant community of users and contributes to collective learning, ultimately benefiting the entire development team and fostering a culture of continuous improvement.
In conclusion, the "Future Enhancements" section outlines numerous possibilities to elevate the "Defect Diary" application's capabilities, user experience, and overall efficiency. By embracing these future enhancements, the application can evolve into a powerful and indispensable tool for defect management, fostering seamless collaboration, data-driven decision-making, and continuous improvement within the software development process.

## XI. CONCLUSION

The "Defect Diary" web application is designed to offer users a practical solution for recording and managing defects along with their corresponding solutions. Its main purpose is to promote knowledge sharing among users, helping them efficiently address similar defects when encountered in the future. By creating a repository of defects and solutions, the project aims to reduce the time and effort required to resolve issues effectively.

The development of this application involved utilizing Asp.net MVC as the framework and C# as the backend language, providing a robust and scalable platform for building the web application. The user interface is intuitive and straightforward, featuring two main sections: "Add Defect" and "Search Defect." Users can easily input new defects by providing necessary details and later search and filter the defect list based on various criteria, including Defect ID, Resolved By, Defect Description, Related Keywords, Application Name, Team Name, Defect Environment, and Resolution Type.

To ensure data accuracy and integrity, the application incorporates various validation measures. These include restrictions on duplicate entries, mandatory fields, and real-time error messages for incorrect or incomplete data. Certain input fields are subject to specific limitations, such as allowing only three-digit numbers in "Estimated Time" and preventing the selection of future dates in "Date Resolved."

Additionally, the application allows for defect editing, provides detailed views of defects by clicking on their IDs from the search results, and ensures smooth navigation between different pages. The "Settings UI" page offers customization options for dropdown menus, ensuring consistency throughout the application.

In conclusion, the "Defect Diary" web application offers an efficient and user-friendly platform for managing defects. Its comprehensive functionality, adherence to best practices, and utilization of Asp.net MVC and C# make it a valuable tool for organizations seeking to streamline defect resolution and knowledge sharing in their software development processes.

## REFERENCES

[1] Ruchika Malhotra and Laavanye Bahl, “A defect tracking tool for open source software”, IEEE 2nd International Conference for Convergence in Technology (I2CT), Mumbai, 2017, doi: 10.1109/I2CT.2017.8226259.
[2] David Russell and Nilesh Patel, “Increasing Software Engineering Efficiency Through Defect Tracking Integration”, IEEE International Conference on Software Engineering Advances (ICSEA'06), Tahiti, French Polynesia, 2006, doi: 10.1109/ICSEA.2006.261261.
[3] A.G. Koru and J. Tian, “Defect handling in medium and large open source projects”, IEEE Software, USA, 2004, doi: 10.1109/MS.2004.12.

DOI: 10.17148/IARJSET.2023.10782

[4] Michael Fredericks and Victor Basili, "Using Defect Tracking and Analysis to Improve Software Quality", DoD Data & Analysis Center for Software (DACS), Maryland USA, 1998.

[5] Olga Baysal, Reid Holmes and Michael W. Godfrey, "Situational awareness: Personalizing issue tracking systems", 2013 35th International Conference on Software Engineering (ICSE), San Francisco USA, 2013, doi: 10.1109/ICSE.2013.6606674.

[6] R. G. Mays, C. L. Jones, G. J. Holloway and D. P. Studinski, "Experiences with Defect Prevention", IBM Systems Journal, 1990, doi: 10.1147/sj.291.0004.

[7] Mika V. Mantyl and Casper Lassenius, "What Types of Defects Are Really Discovered in Code Reviews?", IEEE Transactions on Software Engineering, Finland, 2008, doi: 10.1109/TSE.2008.71.

[8] Jingyue Li, Tor Stalhane, Reidar Conradi and Jan M. W. Kristiansen, "Enhancing Defect Tracking Systems to Facilitate Software Quality Improvement", IEEE Software, Norway, 2011, doi:10.1109/MS.2011.24.

[9] Sandeep Singh, "Analysis of Bug Tracking Tools", International Journal of Scientific & Engineering Research, Amritsar, 2013.

[10] J.N. Johnson and P.F. Dubois, "Issue tracking", IEEE Computing in Science & Engineering, California USA, 2003, doi: 10.1109/MCISE.2003.1238707.