International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 3, May-June 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

BugFixer: A Web-Based Tracking and Project Management System

K. Midhunkumar*, P. Rajapandian**
*(Student, Department of Master Computer Application, Sri Manakula Vinayagar Engineering College, Pondicherry, India
Email: midhunkumar654@outlook.com)
** (Professor, Department of Master Computer Application, Sri Manakula Vinayagar Engineering College, Pondicherry, India
Email: rajapandian.mca@smvec.ac.in)

----------**********----------

Abstract:

Software development teams often face challenges in managing bugs, tracking project progress, and coordinating tasks efficiently. Traditional methods such as spreadsheets, emails, and manual records can result in delayed issue resolution, poor communication, and reduced productivity. To address these challenges, this project presents BugFixer, a web-based Bug Tracking and Project Management System designed to streamline software development workflows. The system provides a centralized platform for managing projects, tracking software bugs, assigning tasks, and monitoring team activities. Users can create projects, report issues, assign tickets, update ticket status, and track progress throughout the development lifecycle. The system supports multiple ticket states including Open, In Progress, Resolved, and Closed, enabling effective issue management and monitoring. In addition to bug tracking, BugFixer includes comment management, activity logs, and project statistics to improve collaboration and transparency among team members. With role-based dashboards for Administrators, Project Managers, Developers, and Submitters, the system ensures secure access and efficient workflow management, ultimately improving productivity, communication, and software quality.

Keywords — Bug Tracking System, Web Application, Software Engineering, Project Management.

----------**********----------

I. INTRODUCTION

Software development projects require effective collaboration among team members to ensure that issues are identified, tracked, and resolved efficiently. As project complexity increases, managing bugs, monitoring progress, and coordinating tasks become more challenging. Many organizations still depend on manual methods such as emails, spreadsheets, or informal communication channels to track software defects. These approaches often lead to delayed issue resolution, poor visibility of project status, duplicated efforts, and communication gaps among team members, ultimately affecting software quality and productivity.

To address these challenges, BugFixer is developed as a web-based Bug Tracking and Project Management System that provides a centralized platform for managing software development activities. The system enables users to create projects, report bugs, assign tickets, monitor issue progress, and collaborate through comments and activity tracking. By maintaining all project-related information in a single environment, BugFixer improves organization, transparency, and workflow efficiency.

The system supports role-based access for Administrators, Project Managers, Developers, and Submitters, allowing each user to perform tasks according to their responsibilities. Through secure authentication, ticket management, project monitoring, and team collaboration features, BugFixer helps development teams streamline issue resolution, improve communication, and deliver

reliable software products in a more efficient and structured manner.

II. LITERATURE SURVEY

Pressman [1] emphasized that effective software project management requires structured methods for tracking defects, managing development activities, and ensuring software quality. His work highlighted the importance of organized workflows and defect management throughout the software lifecycle.

Berander and Andrews [2] discussed the role of bug tracking systems in software maintenance and showed that centralized issue management improves communication, accountability, and defect resolution efficiency among development teams.

Mockus et al. [3] studied large-scale software projects and found that effective defect tracking mechanisms significantly improve software reliability and reduce project delays. Their research demonstrated the importance of monitoring issue history and developer activities.

Fitzgerald and Stol [4] examined collaborative software development practices and identified the need for integrated platforms that support task management, project coordination, and issue tracking within a single environment.

Rahman et al. [5] analyzed modern bug tracking tools and concluded that ticket prioritization, status monitoring, and activity tracking are essential for efficient software maintenance and defect resolution.

Zimmermann et al. [6] investigated defect management practices in software projects and emphasized that systematic bug reporting and tracking contribute to higher software quality and faster issue resolution.

Bird et al. [7] explored collaboration in software engineering and observed that communication among developers plays a significant role in reducing software defects and improving project productivity.

Baysal et al. [8] studied issue tracking repositories and found that maintaining detailed ticket histories and project records helps development teams monitor progress and make informed decisions.

Tsay et al. [9] highlighted the importance of collaborative development platforms that integrate project management, communication, and issue tracking functionalities. Their findings support the development of BugFixer as a web-based Bug Tracking and Project Management System that improves collaboration, transparency, and software quality through centralized project and defect management.

III. THEORETICAL FRAMEWORK

The proposed BugFixer system is grounded in the principles of Software Project Management, Defect Management, and Collaborative Software Development. From the software engineering perspective, Pressman's Software Engineering Framework [1] provides the foundation for managing software defects through structured processes, systematic tracking, and continuous monitoring. The system applies these principles by maintaining centralized records of projects, tickets, comments, and user activities throughout the software development lifecycle.

From the defect management literature, bug tracking theories emphasize the importance of identifying, documenting, prioritizing, assigning, and resolving software issues in a controlled manner. The BugFixer system operationalizes these concepts through ticket creation, status tracking, priority management, and activity monitoring. This structured workflow helps development teams improve software quality while reducing issue resolution time.

From the collaborative software development perspective, modern project management frameworks highlight the need for effective communication and coordination among team members. The system incorporates role-based access control, project assignment, ticket discussions, and activity logs to support collaboration between Administrators, Project Managers, Developers, and Submitters.

Furthermore, the project follows the client-server architecture model, where the React.js frontend communicates with Spring Boot REST APIs and MySQL database services. This architecture supports scalability, maintainability, and secure data management. The theoretical foundation of the

project therefore combines software engineering principles, defect tracking methodologies, and collaborative project management practices to provide an efficient and centralized bug tracking environment.

IV. METHODOLOGY

The development methodology of the BugFixer system follows a structured and iterative software development approach. The process began with a detailed analysis of existing bug tracking and project management practices used in software development organizations. This study identified several limitations in traditional issue management methods, including fragmented communication, manual bug tracking, lack of centralized project monitoring, inefficient task assignment, and limited visibility into project progress.

Following the requirement analysis phase, the system requirements were gathered and categorized into user management, project management, ticket management, comment management, activity monitoring, and reporting functionalities. The requirements were then transformed into system models including use case diagrams, activity diagrams, sequence diagrams, and database design models to establish a clear system architecture.

The implementation phase focused on developing a web-based platform using React.js for the frontend, Spring Boot for the backend, and MySQL for database management. REST APIs were developed to facilitate communication between the frontend and backend components. JWT authentication and Spring Security were integrated to provide secure user authentication and role-based access control.

The system architecture was designed using a layered approach consisting of presentation, business logic, data access, and database layers. This modular architecture improves maintainability, scalability, and system performance. Finally, the developed modules were integrated and tested to ensure proper functionality, reliable bug tracking, effective project management, and smooth collaboration among software development teams.

TABLE I

COMPARISON OF EXISTING SYSTEM VS. PROPOSED SYSTEM

Feature	Existing System	Proposed System
Bug Tracking Method	Manual (spreadsheets, email, chat tools)	Centralized web-based system
Project Management	Separate tools, manual coordination	Integrated project & ticket system
Data Storage	Scattered files and systems	Centralized MySQL database
Ticket Management	Manual reporting & status updates	Structured ticket lifecycle tracking
Status Monitoring	Limited visibility	Real-time status (Open/In Progress/Closed)
Team Collaboration	Email and messaging apps	Built-in comments & activity logs

V. EXISTING SYSTEM

Bug tracking and project management in many software development organizations still rely on a combination of spreadsheets, emails, messaging applications, and standalone tools that operate independently of one another. Developers report issues through emails or shared documents, while project managers manually track task assignments and project progress. This fragmented approach often results in poor coordination, delayed issue resolution, and limited visibility into the overall status of software projects.

The primary challenge is not the lack of project-related information, but the absence of a centralized platform that integrates project management, issue tracking, team collaboration, and activity monitoring. Bug reports, project updates, and developer communications are frequently stored across multiple systems, making it difficult for stakeholders to obtain a complete view of project activities. As projects grow in size and complexity, tracking issue history, monitoring progress, and managing team responsibilities become increasingly difficult.

Existing issue management approaches often depend on manual updates and informal communication channels. These methods can lead to duplicate bug reports, missed assignments, inconsistent status updates, and communication gaps between team members. Furthermore, many organizations lack a structured mechanism for monitoring project activities and maintaining detailed records of issue resolution processes.

User Management	Limited role control	Role-based access (Admin, Dev, PM, User)
Activity Tracking	No central logs	Automated activity logging
Security	Basic login/shared access	JWT authentication & authorization
Reporting	Manual reports	Dashboard analytics
Scalability	Hard to scale	Multi-project scalable system
Project Visibility	Fragmented view	Centralized dashboard

VI. PROPOSED SYSTEM

The proposed system, BugFixer, replaces traditional and fragmented bug tracking approaches with a centralized web-based platform for project management, issue tracking, and team collaboration. The system is designed to provide a structured environment where software development teams can efficiently manage projects, report defects, assign tasks, monitor progress, and maintain complete records of development activities. By integrating multiple development processes into a single platform, the system improves transparency, communication, and overall project productivity.

The core functionality of the system revolves around project and ticket management. Whenever a bug is reported, a ticket is automatically created and assigned a status, priority level, and team member. Throughout the software lifecycle, tickets can be updated, reassigned, and monitored until resolution. Project activities are recorded and made available to authorized users through dashboards. The platform streamlines workflows, enhances accountability, and improves software quality.

A. Text Font of Entire Document

Every user accesses the system through a secure JWT-based authentication mechanism. User credentials are encrypted and securely stored in the database. The system supports role-based access control for Administrators, Project Managers, Developers, and Submitters, ensuring that users can only access functionalities relevant to their responsibilities.

B. Academic Data Collection

The Project Management module enables administrators and project managers to create projects, assign team members, update project information, and monitor project progress. All project-related data is maintained in a centralized database, providing a clear overview of ongoing development activities.

C. AI Risk Prediction Engine

The Ticket Management module serves as the core component of the system. Users can report bugs, define priorities, assign developers, update ticket status, and track issue resolution progress. Each ticket maintains a complete history of modifications, ensuring accountability and traceability.

D. Recommendation & Intervention Engine

The system provides a comment management feature that allows team members to discuss issues directly within tickets. Activity logs automatically record important actions such as ticket creation, status changes, assignments, and project updates, improving collaboration and project visibility.

E. Role-Based Dashboards

Each user is provided with a personalized dashboard based on their role. Administrators can manage users and projects, Project Managers can oversee project activities and assignments, Developers can view and update assigned tickets, and Submitters can create and monitor reported issues. These dashboards provide real-time information and facilitate efficient workflow management across the organization.

VII. SYSTEM ARCHITECTURE

The architecture of the BugFixer system follows a layered, web-based client-server design that ensures scalability, maintainability, and efficient communication between system components. The architecture consists of the Presentation Layer, Application Layer, Database Layer, and Security Layer, which work together to provide project management, bug tracking, and collaboration functionalities.

The Presentation Layer is developed using React.js and provides an interactive user interface

for Administrators, Project Managers, Developers, and Submitters. It contains dashboards, project management screens, ticket management interfaces, comment sections, and statistical reports. The frontend communicates with the backend through RESTful API requests and dynamically updates information based on user actions.

The Application Layer is implemented using Spring Boot and serves as the core business logic layer of the system. It handles user authentication, project management, ticket processing, comment management, activity logging, and system validation. REST APIs are used to exchange data between the frontend and backend while maintaining a clean separation of responsibilities.

The Database Layer utilizes MySQL for storing system information. The database maintains records related to Users, Projects, Tickets, Comments, Ticket History, and Activity Logs. JPA and Hibernate are used to manage database operations and entity relationships efficiently.

The Security Layer incorporates Spring Security and JWT authentication to protect system resources. Passwords are securely encrypted, API requests are authenticated using JWT tokens, and role-based access control ensures that only authorized users can access specific functionalities within the BugFixer system.

This layered architecture improves system scalability, enhances performance, ensures secure authentication, and supports efficient data flow between components, enabling a robust, maintainable, and high-quality BugFixer platform for effective software development management.

[Fig. 1: System Architecture Diagram]

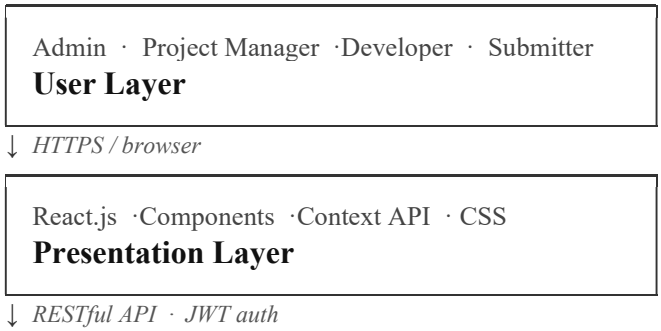

Fig. 1 System Architecture

VIII. SYSTEM MODULES

The BugFixer application is organized around eight functional modules, each with a clearly defined responsibility. The Authentication & Authorization module handles user login, credential validation, session management, and role-based access control. The User Management module maintains user accounts, roles, and project assignments. The Project Management module manages project creation, updates, member allocation, and project monitoring. The Ticket Management module handles bug reporting, assignment, prioritization, and status tracking. The Comment Management module enables collaboration and communication among team members. The Activity Log module records system events and user actions. The Dashboard & Statistics module provides project insights, ticket summaries, and performance indicators. The Notification & Reporting module delivers updates and generates reports for project monitoring and decision-making.

IX. EVALUATION METRICS

The effectiveness of the BugFixer system is evaluated across three dimensions. System functionality is measured by verifying project management, ticket tracking, user management, authentication, and collaboration features against the specified requirements. Special attention is given to ticket creation, assignment, status updates, and issue resolution workflows to ensure that all core operations function correctly and efficiently.

Project management effectiveness is evaluated by monitoring ticket resolution time, project progress tracking, user collaboration, and workflow efficiency. Comparisons with traditional bug tracking methods provide a benchmark for measuring improvements in communication, transparency, and issue management processes.

System performance is evaluated through response time analysis, database query execution, and concurrent user testing to ensure smooth operation under normal workloads. Security effectiveness is assessed through JWT authentication, role-based authorization, and access control validation. User satisfaction is measured through feedback from Administrators, Project Managers, Developers, and Submitters, focusing on usability, reliability, ease of navigation, and overall effectiveness in managing software development projects and bug resolution activities.

X. RESULT AND DISCUSSION

The BugFixer system was successfully developed and tested as a web-based Bug Tracking and Project Management System. Functional testing confirmed that all major modules, including user management, project management, ticket tracking, comment management, activity logging, and role-based access control, operated according to the specified requirements. The system enabled users to create projects, report bugs, assign tickets, update issue status, and collaborate effectively through a centralized platform.

Performance evaluation showed that the application responded efficiently to user requests, with project retrieval, ticket updates, and dashboard operations executing within acceptable response times. The centralized database structure improved data consistency and eliminated the need for manual issue tracking through spreadsheets and emails. Ticket status monitoring and activity logs provided complete visibility into project progress and issue resolution activities.

During usability evaluation, users reported improved collaboration, better project transparency, and easier bug management compared to traditional tracking methods. Project Managers were able to monitor development activities more effectively, while Developers benefited from organized ticket assignments and status tracking. The role-based dashboard design simplified navigation and ensured that users could access relevant information quickly. Overall, the results demonstrate that BugFixer improves software development workflows, enhances communication among team members, and provides an efficient solution for managing projects and software defects.

XI. BENEFITS

The primary benefit of the BugFixer system is the shift from fragmented and manual issue tracking to a centralized and structured software development environment. By providing real-time visibility into project activities, bug reports, and ticket progress, the system enables development teams to identify, prioritize, and resolve software defects more efficiently. Project Managers can monitor project status continuously, while Developers can focus on assigned tasks through a well-defined workflow.

Secondary benefits include improved collaboration through integrated comments and activity logs, reduced administrative effort through automated ticket tracking and status management, and enhanced project transparency through centralized dashboards. The system also improves accountability by maintaining complete records of assignments, updates, and issue resolution activities. Role-based access control ensures secure access to project information while allowing users to perform responsibilities relevant to their roles.

Organizations benefit from improved software quality, faster bug resolution, better resource management, and increased team productivity. By integrating project management, ticket tracking, collaboration, and security within a single platform, BugFixer provides an efficient and scalable solution for managing software development projects and maintaining high-quality software products.

XII. ETHICAL AND PRACTICAL CONSIDERATIONS

The BugFixer system is designed with several ethical and practical considerations to ensure secure, reliable, and responsible management of software development projects. Since the application stores

project information, user accounts, ticket details, and activity records, protecting sensitive data is a primary concern. The system therefore implements role-based access control to ensure that users can only access information and functionalities relevant to their assigned responsibilities.

Data security is enforced through JWT-based authentication, encrypted password storage, and secure API communication. User credentials are never stored in plain text, and access to project resources is restricted based on user roles such as Administrator, Project Manager, Developer, and Submitter. These measures help prevent unauthorized access and maintain the confidentiality of project information.

From a practical perspective, the system is designed to improve collaboration while maintaining accountability. All ticket updates, comments, assignments, and status changes are recorded through activity logs, providing complete traceability of project activities. This helps organizations monitor development progress and resolve disputes when necessary.

To ensure reliability and fairness, the system maintains accurate project records and consistent workflow management. Regular maintenance, database backups, and security updates help protect data integrity and ensure continuous availability of the BugFixer platform for software development teams.

XIII. FUTURE DIRECTIONS

Several directions offer meaningful enhancements to the current BugFixer system. Integration with external development tools such as GitHub, GitLab, and Bitbucket would enable automatic synchronization of repositories, commits, and issue tracking activities, reducing manual effort and improving development workflow efficiency. Real-time notifications through email, mobile applications, or messaging platforms could further improve communication among project stakeholders and ensure timely responses to critical issues.

Advanced analytics and reporting features could provide deeper insights into project performance, developer productivity, bug resolution trends, and software quality metrics. Artificial Intelligence and Machine Learning techniques could also be incorporated to predict bug severity, recommend ticket assignments, and identify potential project risks based on historical project data.

Mobile application support would allow Administrators, Project Managers, Developers, and Submitters to access project information and manage tickets from any location. Multi-organization support could enable the platform to serve multiple companies while maintaining complete data isolation and security. Future enhancements may also include automated testing integration, continuous deployment support, and cloud-based scalability to accommodate larger software development teams and enterprise-level projects.

XIV. CONCLUSION

Software development projects require efficient management of bugs, tasks, and team collaboration to ensure successful project delivery and high software quality. This paper presented BugFixer: A Web-Based Bug Tracking and Project Management System, designed to address the limitations of traditional issue tracking approaches by providing a centralized platform for project management, ticket tracking, collaboration, and activity monitoring.

The system successfully integrates project creation, bug reporting, ticket assignment, status tracking, comment management, and role-based access control into a single environment. Testing results demonstrated reliable system performance, effective project monitoring, and improved collaboration among Administrators, Project Managers, Developers, and Submitters. The React.js frontend, Spring Boot backend, MySQL database, and JWT-based security architecture provided a scalable, secure, and maintainable solution for software development teams.

The primary contribution of this work is the integration of project management and bug tracking functionalities within a unified web-based platform. By improving transparency, communication, accountability, and workflow efficiency, BugFixer helps organizations manage software development activities more effectively while supporting the

delivery of reliable and high-quality software products.

XV. REFERENCES

[1] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill Education, 2020.

[2] P. Berander and A. Andrews,"Requirements prioritization," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Berlin, Germany: Springer, 2005, pp. 69–94.

[3] A. Mockus, R. T. Fielding, and J. Herbsleb, "Two case studies of open source software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 3, pp. 309–346, 2002.

[4] B. Fitzgerald and K. J. Stol, "Continuous software engineering and beyond: Trends and challenges," *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.

[5] F. Rahman, D. Posnett, and P. Devanbu, "Recalling the imprecision of bug severity classifications," in *Proceedings of the 20th ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 2012, pp.1.11.

[6] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," in *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 2009, pp. 91–100.

[7] C. Bird, A. Bachmann, F. Rahman, A. Bernstein, and P. Devanbu, "Fair and balanced? Bias in bug-fix datasets," in *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 2009, pp. 121–130.

[8] O. Baysal, R. Holmes, and M. W. Godfrey, "Developer interactions and collaboration in software engineering," *Empirical Software Engineering*, vol. 18, no. 5, pp. 960–988, 2013.

[9] J. Tsay, L. Dabbish, and J. Herbsleb, "Let's talk about it: Evaluating contributions through discussion in GitHub," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 144–154.

[10] M. Fowler, *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley, 2002.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA, USA: Addison-Wesley, 1994.

[12] R. Johnson, J. Hoeller, J. Arendsen, T. Risberg, and C. Sampaleanu, *Professional Java Development with the Spring Framework*. Indianapolis, IN, USA: Wrox Press, 2005.

[13] R. Johnson, J. Hoeller, J. Arendsen, T. Risberg, and C. Sampaleanu, *Professional Java Development with the Spring Framework*. Indianapolis, IN, USA: Wrox Press, 2005.

