INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 08 ISSUE: 05 | MAY - 2024 SJIF RATING: 8.448 ISSN: 2582-3930

BugSpere (A Bug Tracker for Developer's)

Anurag Mishra
AIT CSE IS
Chandigarh University
India
anurag7706@gmail.com

***Abstract*—** In software development, dependability and quality are critical. A Bug Tracking System (BTS) designed specifically for large-scale software projects is introduced in this significant project with the goal of streamlining bug tracking, reporting, detection, and resolution. The system provides secure authentication, a backend database that is reliable, and an easy-to-use web interface. Using cutting-edge web technologies, it offers critical features including automated notifications, progress tracking, severity assignment, and thorough bug reporting together with user-friendly interfaces for a range of user roles. Tools for data visualization help identify defect patterns and monitor projects. The project follows best practices recommended by the industry for requirements analysis, design, implementation, testing, and deployment, and it solicits ongoing input. The result is a thorough BTS that addresses defect management difficulties. Thorough testing and assessment in comparison to standards confirms its efficacy. This approach could have a big impact on software development, improving user satisfaction and quality. Furthermore, issues with bug tracking are covered, including distributed team communication, scalability, and automation balancing. Proposals for strategies to address these issues are grounded in scholarly research and professional opinions. This summary highlights the critical role that bug tracking systems play in contemporary software development, making it an invaluable tool for researchers, practitioners, and teams looking to improve bug tracking procedures and raise the caliber and effectiveness of the software development lifecycle.

Keywords— Bug Tracking System, Software Development, Defect Management, Quality Assurance, Automation, Collaboration, Large-scale Projects.

I. INTRODUCTION

Alice, a real person: My ECLIPSE crashed.

Bob, a bug-tracking system: What did you do?

Alice: I clicked on File → New Project and OK.

Bob: Did you choose a Java project?

Alice: No... (a few questions later)

Bob: Thanks, Alice. The bug is likely in ProjectCreator.java, and we will fix it soon.

Bug tracking systems are widely used as an organizational tool for maintenance tasks. These tools provide as a central hub for tracking the status of bug reports, getting further details from reporters, and debating possible fixes for the issue. Using the details in bug reports, developers can determine the root cause of a problem and focus on files that are likely to need to be fixed. Developers from the APACHE, ECLIPSE, and MOZILLA projects were surveyed to determine what information items they thought would be helpful in fixing bugs [4]. Developers put information like stack traces, ways to replicate, actual and expected behaviour, test cases, and pictures highly on their list of preferred items.

According to earlier studies, reporters frequently leave out these crucial details [4, 6]. Then, developers are compelled to

actively seek out information from reporters; progress may slow as a result, depending on how responsive the reporters are. This delay causes problems to take longer to fix and causes the number of unresolved defects in the project's bug tracking system to rise. We think that one of the causes of this issue is that the existing bug tracking systems are nothing more than relational databases' interfaces to the reported bugs. They offer reporters little to no assistance in gathering the data that developers require.

In order to address this issue, we are working on ideas that have the potential to significantly improve bug tracking systems' usability and make it easier for most users to collect and report useful information (Section 2). In addition, we provide a decision tree-based simulation of an interactive bug tracking system that helps developers by gathering pertinent user data and using it to locate the file containing the bug (Section 3). Section 4 concludes the paper with a review of related work, and Section 5 offers conclusions.

II. EASE OF USE

1.1 The Problem Definition

Ensuring the high quality and dependability of software products is crucial in the dynamic and always changing field of software development. The successful identification, reporting, monitoring, and resolution of software defects—also known as bugs—is essential to achieving this goal. The lack of a strong bug tracking system can present a number of issues that impede software development teams' general effectiveness and productivity.

Delays in issue resolution, communication breakdowns, and a general degradation in software quality can result from dispersed information and a lack of coordination in the absence of a centralized platform for managing bug reports. Numerous problems could arise from this, such as irate end users, harm to one's reputation, and possible monetary losses.

A bug tracking system becomes an essential tool for managing and keeping track of software faults in order to address these issues. It acts as a centralized platform that makes it easier for developers, testers, and stakeholders to collaborate and communicate effectively, resulting in the prompt and efficient identification, reporting, tracking, and resolution of defects.

Let us examine the case of a software development team working on the establishment of an online store. A wide range of problems can occur during the development process, from wrong pricing and broken links to security flaws and unplanned failures. If these problems are not fixed right away, they may have a negative effect on both the user experience and the website's overall performance.

Key Problem Areas:

The absence of a comprehensive bug tracking system can lead to several key problem areas that hinder effective bug management:

1. Lack of Centralization: Without a centralized repository for bug reports, information becomes scattered across various communication channels, making it challenging to track and prioritize bugs effectively. This can result in critical issues being overlooked while less critical ones receive undue attention.
2. Inefficient Communication: Inadequate communication channels between developers, testers, and stakeholders can lead to misunderstandings, redundant efforts, and delays in bug resolution. A bug tracking system provides a clear and structured platform for communication, ensuring that everyone involved is kept informed of the status of each bug.
3. Unstructured Bug Documentation: Standardized bug reporting processes are often missing, leading to bug reports lacking crucial details such as detailed descriptions, reproduction steps, and expected behaviour. This makes it difficult for developers to accurately understand and resolve bugs efficiently.
4. Lack of Prioritization: Ineffective categorization and prioritization of bugs can result in critical issues being overlooked while less critical ones receive more attention. A bug tracking system should provide mechanisms for prioritizing bugs based on their severity, impact, and potential consequences.

5. Unclear Bug Workflow: The absence of a defined bug resolution workflow can cause confusion regarding the necessary steps following bug reporting. This can lead to delays, inefficiencies, and a lack of accountability in the bug resolution process.
6. Limited Reporting and Metrics: Inadequate tracking mechanisms can make it challenging for managers and stakeholders to monitor the progress of bug resolution and make informed decisions. A bug tracking system should provide comprehensive reporting and metrics to track bug resolution times, identify trends, and measure the effectiveness of bug management processes.
7. Regression Testing Challenges: The connection between bug reports and code changes is often missing, making it difficult to verify whether bug fixes introduce new issues during regression testing. A bug tracking system should provide mechanisms for linking bug reports to code changes, enabling developers to effectively test and verify bug fixes.

OBJECTIVES

- **Efficient Bug Tracking**: The primary objective is to create a platform that enables efficient tracking of software defects, streamlining the entire bug resolution process. This includes the ability to report, assign, monitor, and close bugs seamlessly.
- **Automated Bug Assignment**: Implement a sophisticated algorithm for automated bug assignment based on predefined criteria, reducing manual intervention and ensuring that bugs are routed to the most suitable developers for quick resolution.
- **Seamless Version Control Integration**: Integrate with version control systems like Git and utilize Git hooks or CI/CD tools to track code changes related to bugs. This promotes transparency and collaboration among developers.
- **Enhanced Collaboration**: Foster enhanced collaboration among team members through the built-in messaging system, facilitating real-time communication, file sharing, and code snippet sharing to expedite bug resolutions.
- **Efficient Verification**: Implement automated testing scripts and integrate with Continuous Integration (CI) tools to validate bug fixes automatically. This ensures that bugs are thoroughly tested and verified before closure.
- **Data-Driven Insights**: Develop robust reporting and analytics to provide data-driven insights, allowing project managers to make informed decisions and optimize the development process.
- **Customizable and Scalable Solution**: Create a highly customizable and scalable Bug Tracker that caters to the specific needs and workflows of different development teams. The objective is to ensure that the tool can adapt as projects grow in size and complexity, making it a versatile solution for various organizations.

III. 3. DESIGN FLOW

A. Concept Generation

What is Bug Tracker?

A bug tracker in software development is a tool or system used to manage and track issues or bugs that are found in a software application. It provides a centralized platform for developers, testers, and other stakeholders to report, monitor, and resolve software defects efficiently.

Example - Let's say a software development team is working on building a new web application for an online shopping website. During the development process, various bugs and issues are likely to be discovered by testers and end-users. These issues could range from broken links, incorrect prices, security vulnerabilities, or unexpected crashes.

Features of Bug Tracker

Minimum Viable Product (MVP) Features

1. Bug Reporting:
 - Implement a bug report form in the web application using React.

- Enable users to include code snippets or screenshots in their bug reports.
- Set up a backend server using a programming language like Node.js, and use a web framework like Express to handle form submissions.
- Create a database using tools like MongoDB to store bug report data for CRUD operations (create, read, update, delete).
- Implement server-side validation to ensure the bug report form collects necessary information and prevent spam submissions.

2. Issue Assignment:
- Create a user interface for project managers to manually assign bugs to specific developers using technologies like React.

3. Issue Tracking:
- Design and develop a dashboard or user interface for developers using frontend technologies like React.
- Implement authorization and authentication using Auth0 to restrict access to the bug tracker to authorized users.
- Allow users to view, update, and delete their own bug reports.
- Allow developers to view, update, and delete bugs that have been assigned to them.

4. Collaboration:
- Build a messaging system within the bug tracker using technologies like WebSockets or real-time communication libraries like Socket.IO.

5. Bug Closure:
- Create a user interface for testers to close bugs once they are verified, using frontend technologies like React.
- Implement a background process or scheduler to automatically close bugs after a set period of inactivity using tools like Cron or Celery.

6. Reporting and Analytics:
- Set up a reporting module using data visualization libraries like D3.js or Chart.js, and integrate it into the bug tracker dashboard.
- Utilize data analysis tools like SQL queries to extract meaningful insights from the bug tracker database and generate comprehensive reports and analytics.
- Use backend technologies like Node.js, to handle data processing and serve the reports to the frontend.

Addon Features

1. Bug Reporting:
- Use error tracking libraries like Sentry or LogRocket to capture and log application errors automatically.

2. Issue Assignment:
- Develop an algorithm to automatically assign bugs to developers based on predefined criteria, using Python, Java, or another suitable language.
- Utilize the bug tracker's backend and database to implement the assignment logic and update the bug's assigned developer field accordingly.

3. Issue Tracking:
- Integrate version control systems like Git into the bug tracker to track code changes related to bug fixes.
- Use Git hooks or Continuous Integration/Continuous Deployment (CI/CD) tools to automate the process of updating bug statuses based on code changes and commits.

4. Collaboration:
- Implement features to enable file attachments and code snippets sharing in the messaging system, possibly using file storage services like AWS S3 or Google Cloud Storage.

5. Verification:
- Develop automated testing scripts using testing frameworks like Selenium, Jest, or PyTest to validate bug fixes automatically.
- Integrate Continuous Integration (CI) tools like Jenkins, Travis CI, or CircleCI to automatically run tests whenever code changes are committed to the repository.
- Implement a mechanism in the bug tracker to allow testers to mark bugs as "Verified" when they pass the validation tests.

B. High Level Implementation

In the high-level implementation of the Bug Tracker, we will utilize the MERN (MongoDB, Express.js, React, Node.js) stack to build a robust and scalable system. This stack provides the necessary tools and technologies to develop a full-fledged bug tracking application.

Backend (Node.js and Express.js)

- Server Setup: Set up a Node.js server using Express.js to handle the backend logic of the Bug Tracker.
- Database Integration: Connect to a MongoDB database to store bug reports, user data, and other relevant information.
- Authentication and Authorization: Implement user authentication and authorization using libraries like Passport.js or JWT (JSON Web Tokens) to ensure secure access to the application.
- API Endpoints: Create API endpoints for bug reporting, user management, bug assignment, bug closure, and collaboration features.
- Middleware: Implement middleware for request validation, error handling, and user authentication.
- Real-Time Communication: Integrate WebSockets or a real-time communication library like Socket.IO for instant messaging and collaboration.
- Background Process: Develop a background process or scheduler (using tools like Cron or Celery) to automatically close bugs after a defined period of inactivity.

Frontend (React)

- User Interface Design: Create intuitive and user-friendly interfaces for bug reporting, dashboard, messaging, bug assignment, and bug closure using React components.
- User Authentication: Implement user authentication and registration flows for accessing the Bug Tracker.
- Data Visualization: Utilize data visualization libraries like D3.js or Chart.js to build reporting and analytics interfaces.
- File Upload and Sharing: Enable file upload and sharing features using suitable components and libraries.
- Real-Time Messaging: Develop real-time messaging interfaces using WebSocket or Socket.IO for seamless communication among team members.
- Automated Testing Integration: Integrate automated testing scripts using testing frameworks like Jest for validation of bug fixes.
- Version Control Integration: Connect the Bug Tracker to version control systems like Git, track code changes related to bugs, and automate bug status updates based on code commits.

Deployment and DevOps

- Version Control (Git and GitHub): Use Git and GitHub for version control and collaborative development.
- Continuous Integration/Continuous Deployment (CI/CD): Implement a CI/CD pipeline (e.g., Jenkins, Travis CI, CircleCI) to automate the deployment process.
- DevOps Practices: Follow DevOps best practices to ensure a streamlined and efficient development and deployment workflow.
- Scalability: Design the application to be scalable, allowing it to handle increased workloads as the project grows.
- Data Security: Implement robust data security measures to safeguard sensitive information and user privacy.
- Support and Maintenance: Commit to providing active support and regular updates to ensure the Bug Tracker stays up-to-date with the latest industry trends.

IV. RESULT ANALYSIS

A software development bug tracker is an essential tool for managing and monitoring issues, defects, and improvements in software projects. It helps development teams keep track of reported problems and ensures they are addressed efficiently. The analysis of a bug tracker focuses on its features and the impact it has on the development process and software quality.

Key Features and Their Significance:

1. **Issue Tracking**: The bug tracker allows users to create, view, and manage issues. This includes bug reports, feature requests, and other tasks related to the software. It

ensures that issues are systematically documented and not lost in email threads or chat conversations.

2. **Assignment and Ownership**: Issues can be assigned to specific team members, ensuring clear responsibility for problem resolution. This feature helps in accountability and ensures that issues don't fall through the cracks.

3. **Priority and Severity**: The bug tracker allows for categorizing issues by priority and severity. This is crucial for developers to focus on critical problems first, ensuring that the most important issues are addressed promptly.

4. **Status Tracking**: Tracking the status of issues (e.g., open, in progress, resolved) provides transparency about the progress of bug fixes and feature development. It aids in project management and communication with stakeholders.

5. **Comments and Communication**: Users can add comments, screenshots, and additional information to issues. This feature facilitates collaboration and detailed discussions about problems and their solutions.

6. **Version Control Integration**: Integration with version control systems, such as Git, allows for linking issues to specific code commits. This helps in identifying when and where an issue was introduced, aiding in efficient debugging.

7. **Customization**: The bug tracker can be customized to fit the specific workflow and processes of the development team. This adaptability ensures that the tool complements the team's existing practices.

Impact on Software Development:

1. **Improved Issue Visibility:** The bug tracker provides a centralized location for tracking and managing issues. This means that developers can quickly see the current status of all reported problems, leading to faster response times and issue resolution.

2. **Efficient Resource Allocation:** By categorizing issues by priority and severity, development teams can allocate their resources more efficiently. Critical issues that may impact the software's stability or security can be addressed with urgency.

3. **Enhanced Collaboration:** The tool's comment and communication features facilitate collaboration among team members and stakeholders. This transparent communication streamlines issue resolution and keeps everyone informed.

4. **Continuous Improvement**: Over time, data from the bug tracker can be analysed to identify recurring issues, common sources of defects, and areas of improvement in the development process. This data-driven approach leads to continuous enhancement of software quality.

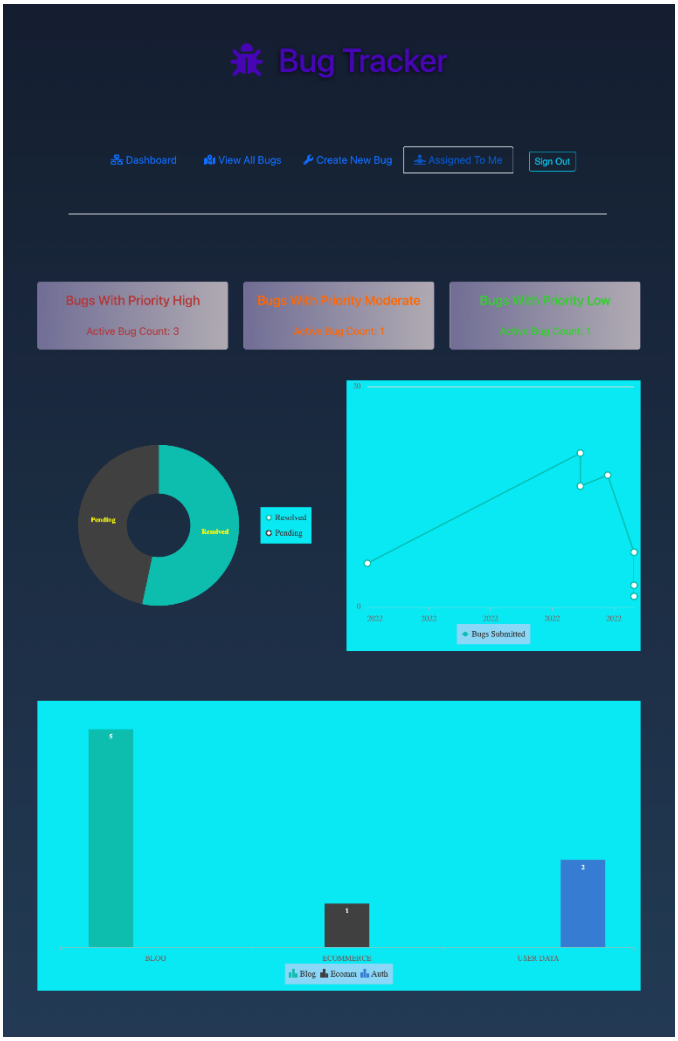
5. **Streamlined Development Workflow:** Integration with version control systems and customizable workflows ensures that the bug tracker fits seamlessly into the development process. It aids in maintaining a structured and efficient workflow.

6. **Satisfied Stakeholders:** Clear and up-to-date information on issue status and progress is beneficial for keeping stakeholders informed and satisfied with the software development process.

Limitations and Future Avenues:

While the bug tracking application presents a significant stride in defect management, it is essential to acknowledge its limitations. The limitations primarily include the scope of implementation, potential technical challenges, and the need for more extensive user testing to validate the system's performance under varying conditions.

Moreover, the future of bug tracking systems lies in the incorporation of artificial intelligence and machine learning to automate bug resolution and predict potential defects. The enhancement of collaboration features and further integration of user experience analysis tools would elevate the application's performance in ensuring a seamless bug management process.



V. CONCLUSION

The bug tracking project presented in this research article serves as a critical stride in addressing the challenges and complexities encountered in managing defects within web-based applications. The development and implementation of an innovative bug tracking application, as detailed throughout this study, underscore the significance of a comprehensive and user-friendly system in ensuring efficient defect management, resolution, and software quality enhancement. This conclusion encapsulates the key findings, insights, and recommendations emerging from our research endeavours.

Recapitulation of Key Findings:

Our study extensively explored the landscape of bug tracking systems, delineating the strengths and limitations of various existing platforms. Through a comparative analysis, it became evident that the bug tracking application developed in this project significantly bridges the existing gaps, offering a unique set of features such as public feedback management, data visualization, and seamless bug resolution. Leveraging the backend and frontend technologies, the system streamlined the bug reporting process, enabling swift identification and resolution of defects encountered across multiple websites.

The innovative integration of Node.js, Express, Mongoose, and React.js provided a robust foundation for the bug tracking application. The implementation of a scalable architecture and the utilization of modern frameworks empowered the system to offer real-time data tracking and responsive frontend design, ensuring an optimal user experience.

Contributions to the Field:

This bug tracking project brings forth notable contributions to the domain of software development and project management. The research successfully introduced a comprehensive application tailored for small to medium-sized teams, enabling streamlined defect tracking and resolution. The provision of customizable and scalable solutions empowers diverse teams to tailor the tool to their specific project needs. Additionally, the data visualization features enhance decision-making capabilities, offering deeper insights into technical stack performances.

Conclusion and Final Remarks:

In conclusion, the bug tracking project outlined in this research article serves as a testament to the growing significance of advanced bug tracking systems in the digital landscape. The comprehensive analysis, development, and implementation of the bug tracking application highlight the system's potential in revolutionizing defect management in web-based

applications. As software quality and efficient project management remain central concerns in the domain, the research offers substantial insights and a foundation for future advancements in bug tracking systems.

REFERENCES

[1] Jungyeon Kim, Geunseok Yang, "Bug Severity Prediction Algorithm Using Topic-Based Feature Selection and CNN-LSTM Algorithm", IEEE Access, vol.10, pp.94643-94651, 2022.

[2] Omar I. Al-Bataineh, Leon Moonen, "Towards Extending the Range of Bugs That Automated Program Repair Can Handle", 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS), pp.209-220, 2022.

[3] Hao Yang, Yang Xu, Yong Li, Hyun-Deok Choi, "K-Detector: Identifying Duplicate Crash Failures in Large-Scale Software Delivery", 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp.1-6, 2020.

[4] Jianjun He, Ling Xu, Yuanrui Fan, Zhou Xu, Meng Yan, Yan Lei, "Deep Learning Based Valid Bug Reports Determination and Explanation", 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), pp.184-194, 2020.

[5] Olga Sokolova, Sergey Kratov, "Platforms for joint development and hosting of software and the example of their implementation in the FAP SB RAS", 2020 International Conference Engineering Technologies and Computer Science (EnT), pp.20-23, 2020.

[6] Sokratis Tsakiltsidis, Andriy Miranskyy, Elie Mazzawi, "On Automatic Detection of Performance Bugs", 2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp.132-139, 2016.

[7] Laud Charles Ochei, Andrei Petrovski, Julian M. Bass, "Implementing the Required Degree of Multitenancy Isolation: A Case Study of Cloud-Hosted Bug Tracking System", 2016 IEEE International Conference on Services Computing (SCC), pp.379-386, 2016.

[8] Yang Xu, Chao Liu, Yong Li, Qiaoluan Xie, Hyun-Deok Choi, "A Method of Component Prediction for Crash Bug Reports Using Component-Based Features and Machine Learning", 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp.773-777, 2023.

[9] Ning Chen, Sunghun Kim, "STAR: Stack Trace Based Automatic Crash Reproduction via Symbolic Execution", IEEE Transactions on Software Engineering, vol.41, no.2, pp.198-220, 2015.

[10] Michael Felderer, Armin Beer, "Using Defect Taxonomies for Testing Requirements", IEEE Software, vol.32, no.3, pp.94-101, 2015.