# School of Computing and Mathematical Sciences

# CO7201 Individual Project

## Preliminary Report

## Software Bug Tracking and Reporting Tool

**Harishmitha Raja**

**hr200@student.le.ac.uk**

**239053031**

**Project Supervisor: Dr. James Hoey**
**Principal Marker: Dr. Newman Lau**

**Word Count: 1646**
**28/02/2025**

**DECLARATION**
All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Harishmitha Raja
Date: 28-02-2025

# Contents

# 1. Aims and Objectives

### 1.1 Motivation

The BugTrackR - Software Bug Tracking and Reporting Tool addresses inefficiencies in bug management that lead to wasted effort, redundant submissions, poor collaboration, and increased maintenance costs [1]. Manual bug assignment, while accurate, is time-consuming, whereas full automation often misallocates due to a lack of contextual decision-making [2]. Duplicate bug reports further burden developers, adding to the backlog and delaying resolutions [3]. Ineffective collaboration worsens these issues, as unstructured communication makes it difficult to track progress [4]. Additionally, as bug report volumes grow, maintaining system performance and usability becomes challenging, further impacting defect resolution [5]. To address these challenges, BugTrackR (Bug Tracking+Reporting) provides a structured, scalable, and efficient workflow that reduces redundancy, enhances collaboration, and enhances bug tracking, ultimately improving software quality and resolution efficiency.

### 1.2 Aims and Objectives

BugTrackR aims to enhance software defect management through a structured, role-based workflow that ensures clear responsibilities, organized issue handling, and efficient collaboration. The application will

- Enable role-based access for secure and structured bug reporting.
- Improve tracking and searchability with status updates, workflows, and advanced filtering.
- Enhance collaboration through structured discussions and real-time notifications.
- Streamline bug prioritization and assignment by balancing manual control with automation.
- Implement duplicate detection to reduce redundancy and improve efficiency.
- Provide role-based analytics for stakeholders, enabling data-driven decision-making for team leads and admins.
- Enhance usability and accessibility with customization options while maintaining a user-friendly experience.

### 1.3 Challenges

- New to AI/ML Techniques for Automation - Implementing AI-driven bug triaging and priority classification requires new skills. To address this, I will study AI/ML concepts, research defect tracking, and experiment with open-source models before integration.

- Limited API & Database Experience - Developing REST APIs, authentication, and MongoDB management is challenging. Solution: Follow best practices, study Node.js (Express) and optimize database design for efficiency.

- Ensuring Security & Access Control - To protect bug data and enforce role-based access, I will implement RBAC, JWT authentication, HTTPS, encryption and pre-deployment security checks will include API security validation, access control, token expiry, and authentication/input validation.

- Balancing Manual & Automated Bug Assignment - Finding the right mix of manual and automated assignment is complex. Solution: Evaluate workload-based, round-

robin, and skill-based assignment, while allowing team leads to configure workload limits.

- Real-Time Collaboration & Notifications - Ensuring live updates for bug tracking and notifications requires reliable technology. I will explore Socket.io for WebSockets and alternative event-driven methods for seamless updates.

## 2. Requirements

**Essential**

- Login & Authentication - The system can allow users to securely log in using their credentials, verify OTP, and reset their passwords if required.
- Role-Based Dashboards - The system can provide users with a dashboard tailored to their role, displaying relevant functionalities and data. (Users can view and track reported bugs, developers/testers can view reported and assigned bugs, update status, add comments, team lead can assign bug, set priority along with the features of developers/testers, admins oversee all applications with full access)
- Bug Reporting
    - General Users - A user can report a bug by providing general details such as browser, device type, and a brief description.
    - Technical Users - A developer, tester, team lead, or admin can report a bug with additional technical details, such as logs, error messages, and replication steps.
- Bug Status Tracking - A user can view and track the status of bugs they have reported.
- Bug Status Updates - A developer, tester, or team lead can update the status of a bug to reflect its current state (Open, Fix in Progress, Fixed, Testing in Progress, Verified, Closed).
- Bug Discussion & Collaboration - A developer, tester, or team lead can comment on bug reports to discuss resolution details.
- Additional Reporter Input - A user can provide additional information after submitting a bug report.
- Manual Bug Assignment - A team lead/admin can manually assign or reassign a bug to a developer.
- Priority Management - A team lead/admin can set and update the priority of a bug (Low, Medium, High, Critical).
- Duplicate Detection - The system can detect and flag potential duplicate bug reports for review by the team lead or admin.
- Email Notifications - The system can send email notifications to users when a bug is assigned, updated with additional information or comments, or its status changes.
- Advanced Search & Filtering - A user can search for and filter bug reports based on various attributes such as status, priority, assigned developer, and date reported.

**Recommended**

- Automated Bug Assignment - The system can automatically assign bugs to developers A team lead or admin can also trigger automated bug assignment manually.
- Automated Priority Classification - The system can automatically assign priority levels to newly reported bugs
- Bug Resolution Analytic - The system can provide role-specific insights, allowing users to track the status of their reported bugs, including those in progress and resolved. Developers and testers can analyze their fix and verification rates, resolution times, and reopened issues, while team leads assess overall team performance.
- Bug Reallocation Request - A developer can request the reassignment of a bug with the ability to provide a reason.
- In-App Notifications - The system can provide real-time notifications within the application for bug status updates, assignments, and approvals.
- Custom Bug Report Export - A user can export bug reports in PDF/CSV format with applied filters.
- Time-Based Alerts for Bug Resolution - The system can trigger automated alerts and UI warnings when bugs remain inactive for a defined period, notifying team leads about unaddressed assignments, testers about pending verifications, and users about unresolved reports to ensure timely resolution.

**Optional**

- GitHub Integration - The system can allow a team lead or admin to create a linked GitHub issue for a bug and automatically sync status updates.
- Customization Options - A user can personalize the application's UI by selecting themes and fonts.
- Favourite Bugs - A user can mark certain bugs as favourites for quick access.
- Unassigned Bug Queue (For Team Assignment) - The system can place bug reports whose issue type is marked as "Other" in an unassigned queue, allowing team leads to manually allocate them to the appropriate team (Frontend, Backend, or DevOps).

## 3. Technical Specification

| Category | Technology | Purpose |
|---|---|---|
| Frontend | React | JavaScript based library for frontend development |
| | HTML5 | For structure and content of web applications |
| | CSS3 | For styling web pages |
| | Tailwind CSS | CSS framework for quick and efficient UI development |
| Backend | Node.js | Used for writing server-side code |
| | Express.js | Framework for creating and managing RESTful APIs |
| Database & ORM | MongoDB | NoSQL database for handling flexible, document-based data storage |
| | Mongoose | Object Data Modeling library for MongoDB to manage data efficiently |
| Authentication & Security | JWT (JSON Web Token) | For secure authentication and user authorization |

| | | |
|---|---|---|
| | SSL | Used for secure data communication between client and server |
| Version Control | GitLab | Platform for version control |
| Real-time Features | Socket.io | Enables real-time features like notifications |
| IDE | Visual Studio Code (VSCode) | Code editor |
| | Sourcetree | GUI tool for simplifying Git operations |
| API Validation | Postman | API testing tool |
| Testing Frameworks | Jest & React Testing Library | Frameworks for unit and integration testing in React. |
| Third-Party Integrations | GitHub API | For integration with GitHub for issue tracking and repository management |
| Task Scheduling | Node Cron | Schedules and automates background tasks at regular intervals. |

# 4. Requirements Evaluation Plan

### 4.1 Evaluation Criteria

The system's effectiveness will be assessed through functionality, usability, security, and performance. Functionality testing ensures core features like bug reporting and tracking work correctly. Usability evaluates ease of navigation for all roles. Security testing detects vulnerabilities, enforcing secure authentication and access control. Performance testing measures responsiveness and system stability for smooth operation

### 4.2 Evaluation Participants

For my project, self-evaluation will be conducted through testing methods, with additional feedback provided by my supervisor during regular review meetings.

### 4.3 Testing

A combination of unit, integration, and system testing will be conducted. Unit tests will validate individual functions and API endpoints, ensuring correctness. Integration testing will verify seamless interaction between different modules, including frontend-backend communication. System testing will assess the overall application, ensuring that all components function correctly together in real-world scenarios.

### 4.4 Verification

The project will be evaluated based on how well it meets the defined requirements, ensuring functionality and usability, passing the planned test cases, and demonstrates stability in different usage scenarios. Supervisor evaluations and structured testing reports will validate the system’s quality and readiness.

# 5. Background Research and Reading list

## 5.1 Background Research

- JIRA [6] and Bugzilla [7] streamline bug tracking with predefined workflows (from one status to another), issue categorization (type, priority), and tracking features (status updates, search and filtering, notifications etc.,). Research on these platforms highlights key functionalities in defect resolution while identifying improvements in bug assignment (categorizing by issue type for team allocation and automating assignments based on workload and developer skills) and tracking reminders (email alerts and UI warnings for unaddressed assignments, verifications and unresolved reports).
- Duplicate bug detection minimizes redundancy by associating similar reports. Research [8] shows that analyzing text descriptions alongside execution logs enhances detection accuracy, improving defect management in BugTrackR.
- Automating bug triage can significantly improve the efficiency of issue resolution by reducing manual effort in assigning bugs. [9] explores how ML models analyze bug descriptions, historical data, and developer expertise to systematically route issues to the appropriate personnel. This research highlights how automation can assist in streamlining the bug triaging process, ensuring that defects are assigned based on relevant factors.
- A user-centered design (UCD) approach ensures intuitive bug tracking by tailoring features like role-based dashboards, advanced search, and streamlined workflows to developers, testers, and team leads. Research highlights that well-structured interfaces enhance navigation and improve ease of use and debugging efficiency. [10].

## 5.2 Reading List

- "MongoDB: The Definitive Guide" by Shannon Bradshaw, Eoin Brazil, and Kristina Chodorow - This book covers NoSQL database principles, indexing, and best practices for handling structured and unstructured data efficiently. [11]

- Node.js & Express.js Official Documentation - Covers backend development, including API creation, authentication (JWT), and middleware for secure and scalable backend logic. [12][13]

- React.js Official Documentation and YouTube Tutorials - Helps in designing a responsive and dynamic frontend for BugTrackR, covering state management, hooks, and performance optimizations. [14][15]

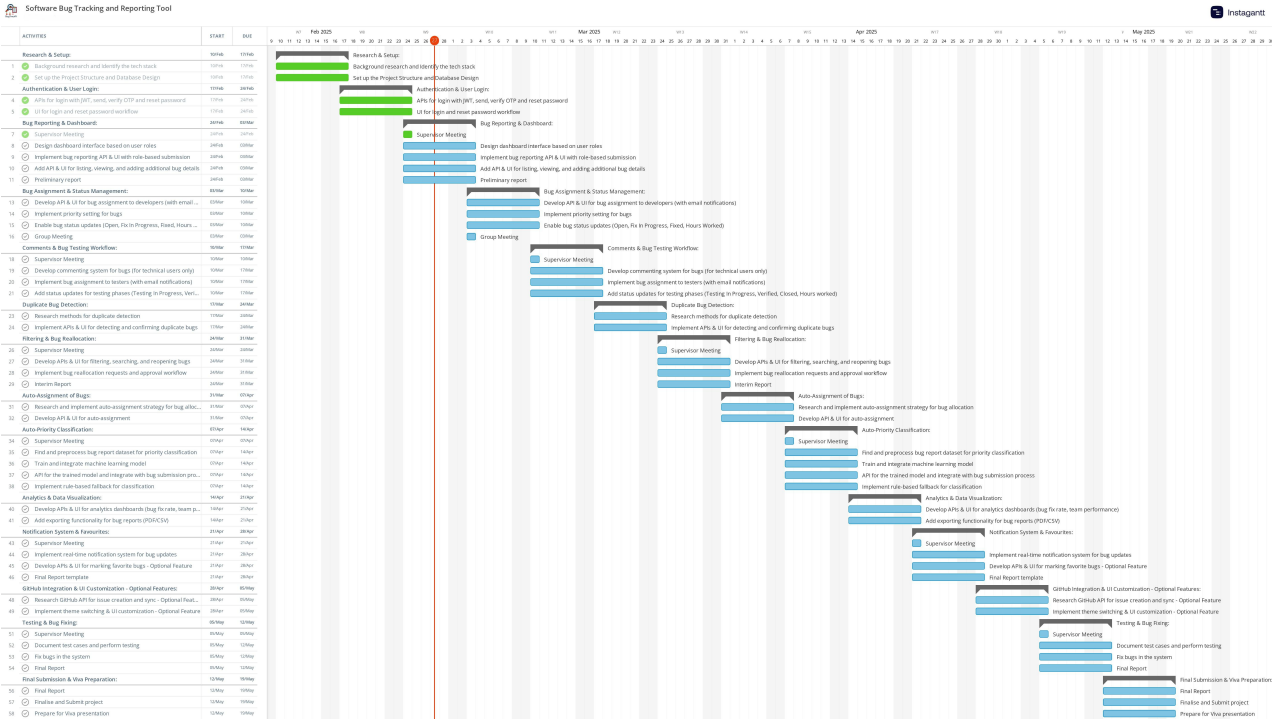
- "Duplicate Bug Report Detection: How Far Are We?" - Discusses textual similarity algorithms, NLP, and machine learning techniques for detecting duplicate bug reports in large-scale bug tracking systems. [16]

- "Automating Bug Triage: Unleashing the Power of Machine Learning and Natural Language Processing" - Explores AI-driven bug triaging techniques to optimize issue assignment using ML models. [9]

# 6. Time-plan and Risk Plan

## 6.1 Time-plan (Gantt Chart)

I have used Instagantt [17] to create a Gantt chart, providing a clear visual representation of the project timeline for better understanding.



### 6.2 Milestones

| Date | Milestone | Description |
|---|---|---|
| 10/02 - 24/02 | Research & Project Setup | Define the tech stack, set up the project structure, and implement authentication with role-based access. |
| 24/02 - 24/03 | Core Bug Reporting & Management | Develop bug reporting, tracking, dashboards, manual assignment, and priority manageament. |
| 24/03 - 07/04 | Advanced Bug Handling | Implement duplicate detection, bug filtering, reallocation requests, and automated bug assignment. |
| 07/04 - 21/04 | ML-Based Enhancements & Role-Based | Integrate ML for priority classification, develop role-based analytics, and enable bug report exports. |
| 21/04 - 05/05 | Notifications & Integrations | Implement email and in-app notifications, integrate GitHub API, and add UI customization. |
| 05/05 - 19/05 | Testing, Optimization & Final Submission | Conduct comprehensive testing, bug fixes, final report submission, and viva preparation. |

### 6.3 Risk Plan

- To address unfamiliarity with new technologies like machine learning and GitHub API integration, time will be allocated for research, prototyping, and fallback strategies.
- API and UI integration risks will be mitigated by validating features regularly to ensure stability and functionality.
- Potential database modifications will be handled by designing a suitable schema upfront to accommodate future changes.
- To prevent unauthorized access to bug reports and system breaches, the system will implement JWT authentication, role-based access control (RBAC), and secure API endpoints.
- Database backups and version control will be maintained to prevent data loss from system failures.
- The system will use a combination of email and in-app notifications, ensuring reliability by allowing in-app alerts to serve as a backup if emails fail and vice versa.

## 7. References


[1] Zimmermann, T., Premraj, R., Bettenburg, N., Just, S., Schroter, A. and Weiss, C. What Makes a Good Bug Report? IEEE Transactions on Software Engineering, 36(5), pp.618–643. https://doi.org/10.1109/tse.2010.63

[2] Bhattacharya, P., & Neamtiu, I. "Fine-Grained Incremental Learning and Multi-Feature Tossing Graphs to Improve Bug Triaging." Proceedings of the IEEE International Conference on Software Maintenance (ICSM), 1–10. https://dl.acm.org/doi/10.1109/ICSM.2010.5609736

[3] Bettenburg, N & Premraj, R. & Zimmermann, T & Kim, S. Duplicate Bug Reports Considered Harmful ... Really? IEEE International Conference on Software Maintenance, ICSM.337-345.10.1109/ICSM.2008.4658082. https://www.researchgate.net/publication/224343297_Duplicate_Bug_Reports_Considered_Harmful_Really

[4] Hooimeijer, P., & Weimer, W. "Modeling Bug Report Quality." Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE), 34-43. https://dl.acm.org/doi/10.1145/1321631.1321639

[5] Mockus, A., & Herbsleb, J. D. "Expertise Browser: A Quantitative Approach to Identifying Expertise." Proceedings of the 24th International Conference on Software Engineering(ICSE),503-512. https://www.researchgate.net/publication/2543802_Expertise_Browser_A_Quantitative_Approach_to_Identifying_Expertise

[6] "Jira Documentation. Atlassian Support. Atlassian Documentation." https://confluence.atlassian.com/jira

[7] "Features." Bugzilla, 2025, www.bugzilla.org/about/features.html#localization. Accessed 27 Feb. 2025.

[8] Wang, Xiaoyin, et al. "An Approach to Detecting Duplicate Bug Reports Using Natural Language and Execution Information." Proceedings of the 13th International Conference on SoftwareEngineering-ICSE'08,2008, https://www.researchgate.net/publication/221554559_An_approach_to_detecting_duplicate_bug_reports_using_natural_language_and_execution_information

[9] Automating Bug Triage: Unleashing the Power of Machine Learning and Natural Language Processing. https://www.researchgate.net/publication/382455930_Automating_Bug_Triage_Unleashing_the_Power_of_Machine_Learning_and_Natural_Language_Processing

[10] Kualitee. User-Centric Bug Resolution: Best Practices with Bug Tracking Software. [Online]. Available at: https://www.kualitee.com/blog/bug-management/user-centric-bug-resolution-best-practices-with-bug-tracking-software/

[11] Bradshaw, S., Brazil, E., & Chodorow, K. MongoDB: The Definitive Guide. Available at: https://librarysearch.le.ac.uk/discovery/fulldisplay?docid=alma991009031379702746&context=L&vid=44UOLE_INST:VE&lang=en&search_scope=MyInst_and_CI&adaptor=Local%20Search%20Engine&tab=Everything&query=any,contains,MongoDB:%20The%20Definitive%20Guide&pfilter=rtype,exact,books&offset=0

[12] Node.js Official Documentation. Introduction to Node.js. Available at: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

[13] GeeksforGeeks. Express.js Guide. Available at: https://www.geeksforgeeks.org/express-js/

[14] React.js Official Documentation. Your First Component. Available at: https://react.dev/learn/your-first-component

[15] The Net Ninja. React.js Crash Course (YouTube Series). Available at: https://www.youtube.com/watch?v=j942wKiXFu8&list=PL4cUxeGkcC9gZD-Tvwfod2gaISzfRiP9d

[16] Bettenburg, N., Just, S., Schröter, A., Weiss, C., Premraj, R., & Zimmermann, T. Duplicate Bug Report Detection: How Far Are We? Available at: https://dl.acm.org/doi/10.1145/3576042

[17] “Instagantt. Online Gantt Chart Software. Asana Integration.” https://app.instagantt.com/