**A picture containing text

Description automatically generated**

**School of Computing and Mathematical Sciences**

**CO7201 Individual Project**

**Preliminary Report**

**[Title of your project]**

**[Your name]**

**[Your email address]**

**[Your student ID]**

**Project Supervisor: [XXXX]**

**Principal Marker: [XXXX]**

**Word Count: [XXXX]**

**[Submission Date]**

**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: [Your name]

Date: [Date of submission]

Contents

[**1.** **Aims and Objectives** 3](#_Toc65510906)

[**2.** **Requirements** 3](#_Toc65510907)

[**3.** **Technical Specification** 3](#_Toc65510908)

[**4.** **Requirements Evaluation Plan** 3](#_Toc65510909)

[**5.** **Background Research and Reading list** 3](#_Toc65510910)

[**6.** **Time-plan and Risk Plan** 3](#_Toc65510911)

[**7.** **References** 3](#_Toc65510912)

# **Aims and Objectives**

**1.1 Motivation**

The “BugTrackR - Software Bug Tracking and Reporting Tool” is developed to address inefficiencies in bug management that often lead to wasted effort, as developers spend time navigating disorganized reports, handling redundant bug submissions, and poor collaboration, ultimately resulting in delays and increased maintenance costs [1]. Ineffective bug assignment further contributes to these challenges, as fully manual allocation, though accurate, is time-consuming, while fully automated assignment often lacks contextual decision-making, leading to misallocation of resources [2]. Additionally, redundant bug reports create unnecessary workload for developers, as a significant portion of reported defects are duplicates, which add to the backlog and slow down resolution times [3]. Collaboration gaps make these inefficiencies worse, as scattered discussions and unstructured communication make it difficult for teams to coordinate effectively and track issue progress [4]. As bug report volumes increase, maintaining performance and usability in tracking systems becomes increasingly difficult, leading to further inefficiencies in defect resolution[5]. Addressing these challenges requires a system that provides a structured and efficient workflow for bug tracking, minimizes redundant work, enhances collaboration, and adapts to scaling demands, ultimately improving software quality and defect resolution efficiency.

**1.2 Aims**

* Develop a structured bug management system where user roles determine access levels and responsibilities, ensuring accurate reporting and organized issue handling.
* Enable bug reporters to track the progress of reported and assigned bugs through clear status updates and organized workflows, and efficient search and filtering capabilities.
* Improve collaboration and communication by enabling structured discussions within bug reports, ensuring that relevant stakeholders can efficiently contribute to issue resolution.
* Enhance bug prioritization and assignment by implementing a hybrid approach that combines manual and automated distribution, balancing control with efficiency.
* Provide role-based analytics, allowing all stakeholders to view relevant data while enabling team leads and admins to monitor performance and make data-driven decisions.
* Improve usability and accessibility by offering customization options, ensuring adaptability for different users while maintaining a user-friendly experience.

**1.3 Objectives**

The objective of this application is to improve software defect management by providing a structured and collaborative workflow for reporting, tracking, and resolving bugs efficiently. The system integrates manual and automation to optimize bug assignment, ensuring balanced workload distribution while maintaining human judgment where necessary. It minimizes redundant efforts by identifying and flagging duplicate bug reports, reducing backlog and improving resolution times. Role-based access and centralized discussions enhance collaboration by keeping developers, testers, and team leads aligned throughout the bug lifecycle. Additionally, advanced search, filtering, and analytics enable stakeholders to track issue progress and make data-driven decisions. By ensuring an organized and adaptable approach to bug tracking, BugTrackR enhances team coordination, streamlines defect managment, and contributes to overall software quality improvement.

**1.4 Challenges**

1. New to AI/ML Techniques for Automation

Challenge: Implementing automated bug triaging, duplicate detection, and priority classification may require AI/ML-based techniques, which I am unfamiliar with. Understanding dataset usage, training models, and integrating them effectively into the system poses a learning curve. Solution: To mitigate this, I will study AI/ML fundamentals related to text classification, explore existing research on defect tracking automation, and experiment with open-source models before integrating them. Additionally, I will seek guidance from research papers, online courses, and AI/ML documentation to ensure a practical approach.

2. Limited Experience in API Development & Database Management

Challenge: Developing a backend with API endpoints for bug management, user roles, notifications, and external integrations (e.g., GitHub) requires expertise in REST API design, authentication, and MongoDB database management, which I have limited experience with.

Solution: I will follow API development best practices, study official documentation for frameworks like Node.js (Express). Additionally, I will explore MongoDB schema design, indexing, and query optimization to ensure efficient data handling.

3. Ensuring Data Security & Access Control

Challenge: Securing bug reports, user roles, and discussions is crucial to prevent unauthorized access and security vulnerabilities. SolutionI will implement role-based access control (RBAC), JWT authentication, and encryption while ensuring secure API communication (HTTPS, CORS). Regular security audits and testing for common vulnerabilities (SQL Injection, XSS, CSRF) will be conducted.

4. Hybrid Bug Assignment Implementation

Challenge: Ensuring a fair and efficient balance between manual and automated bug assignment while determining optimal workload-based auto-assignment criteria.

Solution: I will evaluate priority-based, round robin and skill-based strategies to select the most suitable approach. Team leads will have configurable options to enable or disable auto-assignment and set workload limits to prevent overloading developers.

5. User Experience Design for Different Roles

Challenge: Designing an intuitive interface that caters to the diverse needs of users, developers, testers, team leads, and admins can be challenging. Each role requires different levels of access, workflows, and data visibility, making it difficult to create a seamless experience for all users.

Solution: I will implement role-based dashboards that display only relevant information and actions for each user type, reducing complexity and improving navigation. UI consistency will be maintained through standardized layouts, clear labeling, and structured workflows, ensuring ease of use across all roles.

6. Handling Real-Time Collaboration & Notifications

Challenge: Implementing real-time updates for bug status changes, comments, and notifications requires technologies like WebSockets or event-driven architectures, which I have limited experience with. Ensuring reliable real-time updates without performance issues is a challenge. Solution: I will explore WebSockets using Socket.io in JavaScript for real-time updates on bug status changes and comments. Additionally, I will research other event-driven methods, including polling or microservices, to ensure seamless real-time communication.

# **Requirements**

List of requirements, including high-level requirements or aims, and detailed requirements or objectives. You should separate these into essential, recommended and optional.

# **Technical Specification**

Technical specifications of the project.

# **Requirements Evaluation Plan**

**4.1 Evaluation Criteria**

The system's effectiveness will be assessed through multiple quality attributes. Functionality testing will ensure core features such as bug reporting, tracking, assignment, and status updates operate correctly. Usability focuses on ease of navigation and efficiency for all user roles. Security checks will identify vulnerabilities such as unauthenticated and unauthorized access, ensuring data protection through secure authentication and authorization mechanisms. Performance testing will evaluate responsiveness and system stability to ensure smooth operation

**4.2 Evaluation Participants**

Participants will include individuals acting as developers, testers, tech leads, admins, and end users, ensuring the system meets the needs of both technical and non-technical users. Academic reviewers will provide feedback on the system’s effectiveness, completeness, and adherence to software quality standards.

**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 considered successful if it meets all defined functional requirements, passes all planned test cases, and demonstrates stability in different usage scenarios. Supervisor evaluations and structured testing reports will validate the system’s quality and readiness.

# **Background Research and Reading list**

**5.1 Background Research**

* JIRA and Bugzilla are widely used bug tracking systems that provide predefined processes, issue categorization, and tracking features. These platforms help development teams manage software defects efficiently by organizing reports, assigning issues, and maintaining issue histories. Research into these platforms highlights best practices in defect resolution and workflow coordination, while also identifying areas where alternative approaches to bug assignment, duplicate detection and issue tracking reminders have been explored to improve efficiency.
* Detecting duplicate bug reports is essential for reducing redundancy and improving efficiency in issue tracking. An Approach to Detecting Duplicate Bug Reports Using Natural Language and Execution Information explores how analyzing both textual descriptions and execution data can help identify similar bug reports. By leveraging these techniques, BugTrackR can streamline defect management, ensuring that related issues are linked, reducing duplication, and enhancing the overall bug resolution process.
* Automating bug triage can significantly improve the efficiency of issue resolution by reducing manual effort in assigning bugs. [6] 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 that bug tracking systems are intuitive and tailored to the needs of developers, testers, and team leads. Features like customizable dashboards, advanced search and filtering, and streamlined workflows improve navigation and efficiency. A centralized system further enhances collaboration by allowing teams to track issues in real time, reducing delays and improving issue resolution. Research on UCD principles in defect tracking systems emphasizes that well-structured interfaces contribute to faster response times and a more effective debugging process [2].

**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. [1]

• Node.js & Express.js Official Documentation - Covers backend development, including API creation, authentication (JWT, OAuth), and middleware for secure and scalable backend logic. [2][3]

• React.js Official Documentation and YouTube Tutorials - Helps in designing a responsive and dynamic frontend for BugTrackR, covering state management, hooks, and performance optimizations. [4][5]

• "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. [6]

• "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. [7]

# **Time-plan and Risk Plan**

A detailed timetable and plan for achieving the objectives of the project (this could be tabulated or in the form of a Gantt chart), including the milestones of the project and a risk plan; Explain target dates and amount of time required for the completion of aspects of the project

# **References**

The reference list should contain a mixture of books, research papers (if appropriate) and internet resources.

Zimmermann, T., Premraj, R., Bettenburg, N., Just, S., Schroter, A. and Weiss, C. (2010). What Makes a Good Bug Report? IEEE Transactions on Software Engineering, 36(5), pp.618–643. doi:https://doi.org/10.1109/tse.2010.63

Bhattacharya, P., & Neamtiu, I. (2010). "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

Bettenburg, N., Premraj, R., Zimmermann, T., & Kim, S. (2008). "Duplicate Bug Reports Considered Harmful? A Case Study of Firefox Developers." Proceedings of the IEEE International Conference on Software Maintenance (ICSM), 337-345.

Bettenburg, Nicolas & Premraj, R. & Zimmermann, Thomas & Kim, Sunghun. (2008). 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

Hooimeijer, P., & Weimer, W. (2007). "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

Mockus, A., & Herbsleb, J. D. (2002). "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>

Kualitee (2024). User-Centric Bug Resolution: Best Practices with Bug Tracking Software. [Online]. Available: <https://www.kualitee.com/blog/bug-management/user-centric-bug-resolution-best-practices-with-bug-tracking-software/>

[1] Bradshaw, S., Brazil, E., & Chodorow, K. MongoDB: The Definitive Guide. Available at: https://librarysearch.le.ac.uk/discovery/fulldisplay?docid=alma991009031379702746

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

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

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

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

[6] 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

[7] Automating Bug Triage: Unleashing the Power of Machine Learning and Natural Language Processing. Available at: https://www.researchgate.net/publication/382455930\_Automating\_Bug\_Triage\_Unleashing\_the\_Power\_of\_Machine\_Learning\_and\_Natural\_Language\_Processing