AI-Powered Software Bug Tracking Tool

Ahmad Haizar bin Sahmat
Faculty of Computer Science & Technology
University of Malaysia, Kota Samarahan, Sarawak
ahmd.haizar@gmail.com

Dr Tan Ping Ping
Faculty of Computer Science & Technology
University of Malaysia, Kota Samarahan, Sarawak
pptan@unimas.my

2025 IEEE International Conference on Computing (ICOCO) | 979-8-3315-7539-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICOCO67189.2025.11334082

***Abstract*—Most information technology (IT) companies still rely on manual approaches to manage and track software bugs during testing sessions. This traditional method is time-consuming, error-prone, and lacks efficiency particularly in large or complex software projects. To address these challenges, this paper proposes an artificial intelligence AI-powered Software Bug Tracking Tool designed to streamline the process of reporting, tracking, categorizing, and prioritizing bugs. By leveraging Artificial Intelligence (AI) technologies, including machine learning, the system can automatically classify bugs based on their severity, providing valuable insights for better decision-making. Additionally, the system employs unsupervised learning techniques to detect patterns and relationships among bugs. An OpenAI GPT-4o model, accessed via the Azure OpenAI Service, is integrated into the system to automatically generate summaries of bug reports, helping team quickly understand the key details. A user-friendly design is also provided to support collaboration between developers and testers, reducing the learning curve and improving overall software quality. For bug severity detection, the F1-score is 0.64 when detection Critical bugs; an indicator that the approach might be useful. Similarity bug detection is more challenging because the similarity scores might not necessarily reflect the bug similarity. Test case summarization is based on the OpenAI API. The implementation of this system aims to enhance team productivity, ensure comprehensive bug tracking, and align with modern software development practices. Based on the usability testing with 10 participants, including software expert from various background such as software testers and developers, the duplicate detection and bug summarization features are particularly useful, to streamline the bug tracking process and improve the understanding of reported issues. However, the bug severity prediction feature produced unexpected classifications in certain cases, indicating the need for additional training data and the potential future works to enable the model to continuously improve over time.**

Keywords—Artificial intelligence, Bug Tracking, Machine Learning, Software Testing

I. INTRODUCTION

The testing phase plays a crucial stage in any Software Development Lifecycle, as it ensures the product is free of software bugs and meet all requirements before delivery. During this phase, software development and testing team collaborate to identify, analyze and resolve software bugs within the system. Most tech companies nowadays rely on open-source bug tracking tools which available from certain provider to keep track of software bugs throughout testing. However, these existing tools often use traditional bug reporting approach, requiring testers and developers to analyze software bugs manually such as predicting severity, writing summaries of the bug, and identifying relationships between logged bugs. This manual approach can be time-consuming and prone to inconsistencies, which not only reduce the efficiency and accuracy of bug reporting and resolution but also leads to project time extensions. Therefore, by integrating AI in software bug tracking tool, development and testing teams can identify, analyze and resolve bugs more effectively, leading to improved productivity and software quality.

II. MATERIALS & METHODS

A. Unified Appproach

This project was implemented using one of the methodologies that leveraging object-oriented principles within the Unified Process framework, which is Unified Approach methodology (UA). UA is a software development methodology that follows an iterative and incremental approach to develop systems [1]. It divides the development lifecycle into four phases: Inception, Elaboration, Construction, and Transition, each with specific objectives and milestones. Fig. 1 shows the Unified Approach methodology and its phases.

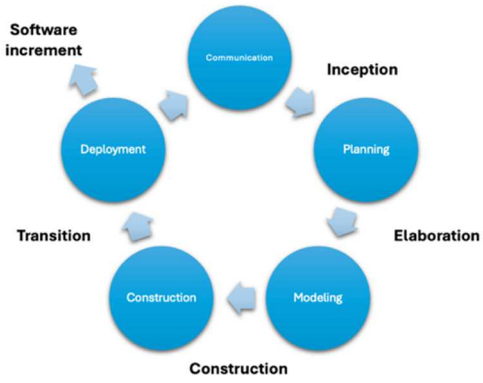

Fig. 1. Unified Approach Methodology

The Unified Approach methodology structured framework ensured that all functional and non-functional requirements were accurately obtained and aligned with stakeholder expectations. By integrating this methodology into the planning phase, the process established a clear foundation for analyzing requirements and translating them into a cohesive system design. During this phase, interviews were conducted to identify the functional and non-functional needs of the project (as shown in Table I). The functional requirements were also obtained by analyzing existing tools to ensure that the proposed system aligns with current market standards and user expectations. By combining these insights with stakeholder input, the requirements were tailored to form a comprehensive foundation for the system design.

TABLE I. REQUIREMENTS SPECIFICATIONS OF THE SYSTEM

Req. No.	Requirement Description
REQ-1	Administrator shall be able to login and redirect to dashboard
REQ-2	User should be able to register their own account
REQ-3	Team member shall be able to login and redirect to specific dashboard
REQ-4	Users shall be able to assign their own role

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on January 29,2026 at 00:02:04 UTC from IEEE Xplore. Restrictions apply.

Req. No.	Requirement Description
REQ-5	Users shall have access to the relevant modules associated with their role after logging in
REQ-6	Users should be able to update their profile
REQ-7	Users shall be able to delete their account
REQ-8	The AI shall enhance and suggest additional relevant details based on the initial bug description provided by the user
REQ-9	The system shall allow user to accept, reject, or modify AI-generated suggestions before finalizing the bug description
REQ-10	The Machine Learning model shall analyze the bug description and assign a severity level (e.g., Minor, Major, Critical etc.)
REQ-11	The system shall display the predicted severity level to the user
REQ-12	The system shall continuously improve the accuracy of severity predictions by learning from user over time
REQ-13	The Machine Learning model shall analyze bug details to detect potential relationships (e.g., duplicates, dependencies, or root causes)
REQ-14	The Machine Learning model shall learn from user feedback on identified relationships to continuously improve its accuracy in finding related bugs.
REQ-15	Team member shall be able to manage bug (Create, View, Update and Delete) details
REQ-16	Team member shall be able to manage (Create, View, Update and Delete) project details
REQ-17	Team member shall be able to manage (Create, View, Update and Delete) project team
REQ-18	The system shall allow a project to be divided into sprints
REQ-19	Team member shall be able to view assigned projects
REQ-20	Team member shall be able to update the status of the reported bug
REQ-21	Team Member shall be able to search reported bug
REQ-22	The system shall allow team member to comment and add attachments to bug discussion
REQ-23	The system shall allow team members to receive notification upon status of a bug is changed
REQ-24	The system shall allow team member to generate reports and insights for each project and phases
REQ-25	The system shall allow team member to move bugs from one sprint to another
REQ-26	The system shall allow team member to generate reports and insights for each project, team and sprints

B. *Comparison to Existing Software Bug Tracking Tools*

This section examines existing software bug tracking tools to identify relevant software technologies, and highlights the challenges and opportunities in the field. By understanding these aspects, the review aims to inform the design and implementation of the proposed tool to ensure it meets current industry standards and addresses the existing gaps (Table II). Additionally, reviews on research works are carried out to provide a clearer understanding of the overall proposed project.

TABLE II. COMPARISON TABLE OF EXISITING TOOLS

Criteria	Excel	Mantis	Jira	Clickup
User interface	Widely recognized for its simplicity and familiarity	Outdated User Interface	Modern design but with complex user interface	Modern User Interface design
Notification	No	Built-in notification system is provided		
Bug Reporting	Bug reporting relies heavily on manual entry	Some of the bug details relies on manual entry	Some of the bug details are automatically generated	Requires less manual entry as it is mostly automated but does not include bug severity information
Bug Analysis	No	Able to analyze software bugs, but user intervention is required		AI chatbot is provided to assist in conducting bug analysis
Collaboration	Poor	Support real-time collaboration		
AI integration	No		Yes	

C. *Artificial Intellegence and Machine Learning Integration*

1) Bug severity prediction

The bug severity prediction feature is developed using a supervised machine learning approach. Multinomial Logistic Regression algorithm [2] is selected to train the model on historical bug report dataset [3]. Each report is represented using numerical features extracted from the text which is the bug description, and the model learns to classify the severity level (e.g., critical, major, minor) based on the input. Fig. 2 shows the full process of building the bug severity prediction model.

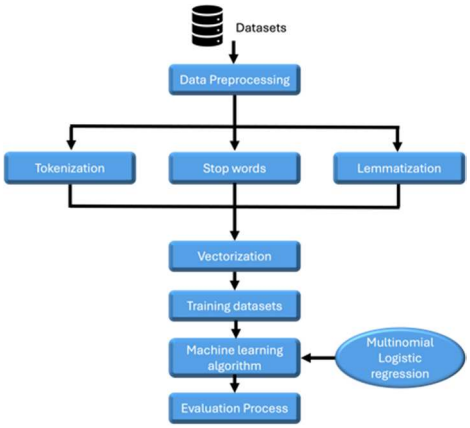

Fig. 2. The bug severity prediction model

Dataset

Kaggle provide an open-source datasets and available for all users to use. In this project, a dataset titled “Bugzilla Bug Reports” [3] will be used to train and test the machine learning model. This dataset consists of 50,000 bug reports from the Bugzilla project. It has been widely used by developers, researchers, and practitioners for various projects and research purposes.

Data preprocessing

Natural language processing (NLP) techniques are applied to the data preprocessing, such as lowercasing, stopword removal, and lemmatization to prepare the textual data before vectorization [4].

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on January 29,2026 at 00:02:04 UTC from IEEE Xplore. Restrictions apply.

Vectorization

The cleaned bug descriptions were converted into numerical form using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique [5] to capture the importance of words across the dataset.

Machine Learning Model

A Logistic Regression model was trained to classify the severity levels (e.g., Minor, Major, Critical) based on the vectorized text features. The dataset was split into training and testing sets to evaluate the model's generalizability. During this phase, it was observed that the dataset had an imbalance in the distribution of severity classes, which could lead to biased predictions. In order to address this, the SMOTE (Synthetic Minority Over-sampling Technique) method [6] was applied to the training data. This technique synthetically generates new instances of the minority class to balance the dataset, thereby improving the ability of the model to learn from all classes without bias.

Evaluation Process

Evaluation metrics such as accuracy, precision, recall, and F1-score were calculated to assess the model's performance [7]. Fig. 3 shows the confusion matrix of each class. Table 3 showed the comparative results for the model on each target (Minor, Major, Critical).

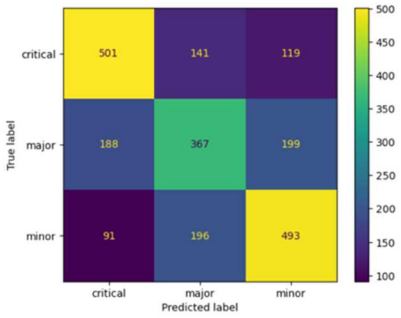

Fig. 3. Confusion matrix for bug severity prediction

2) Bugs Similarity

Vectorization

In this feature, both the bug title and description are concatenated to form a single textual input for each bug report. This combined text is then vectorized using the CountVectorizer technique, a common approach in natural language processing for converting text into numerical format [4]. Fig. 4 shows the vectorization process of the bug description.

```
[661]: from sklearn.feature_extraction.text import CountVectorizer
       vectorizer = CountVectorizer(stop_words='english')
       vector = vectorizer.fit_transform(df['tags'].values.astype('U')).toarray()
       df['tags']

[661]: 0    expose a small  simple ui for 'undo closed tab...
       1    undo close tab (a way to quickly reopen a clos...
       Name: tags, dtype: object
```

Fig. 4. Vectorization process for bug description

Similarity computation

Cosine similarity is a metric used to measure the similarity between two vectors by calculating the cosine of the angle between them [8]. The resulting value ranges from 0 to 1, where 0 indicates no similarity and 1 indicates that the vectors are identical in direction. In the context of this project, cosine similarity is used to determine how similar two bug reports are based on their textual content (e.g., title and description) [6]. Table IV presents the computed similarity scores between reported bugs, providing insight into how closely each pair of bug reports is related in terms of their semantic meaning.

Evaluation process

In order to evaluate the performance of the bug similarity detection feature, average score of the duplicate bugs. A threshold of 0.502 is established as the acceptance score. This means that any similarity scores equal to or greater than 0.502 will be considered an indication of similar or potentially duplicate bug reports, while scores below 0.502 will be classified as not similar or unrelated. To overcome this issue, stating model confidence score in the similar reported bug section might boosts user trust and willingness to accept recommendations.

However, the decision to mark issue as duplicate is still up to the user. The similarity feature is only meant to suggest related issues that might be duplicates, but the decision is up to the user whether they truly are or not.

3) Bug Summarization

To implement the bug summarization feature, the Azure OpenAI service is used to generate concise and readable summaries of reported issues. The system integrates with the Azure OpenAI API by sending bug content along with a carefully designed prompt. The GPT-4o model is specified in the API call to perform the summarization task, due to its strong language understanding and summarization capabilities. Fig. 5 shows the prompt design for bug summarization feature.

```
Summarize the issue by capturing the main problem, its root cause, and important activities, while keeping it short
Title: {$issueTitle}
Created date: {$issueCreatedAt}
Description: {$issueDesc}
Root Cause: {$issueRootOfCause}
Activities Log: {$combinedLog}
```

Fig. 5. GPT Prompt Design

The summarization process is guided by three core components: main problem, root cause and activities log, which condenses the sequence of actions, investigations, and attempted solutions. The inclusion of the activities log is particularly important, as it captures the dynamic progression of the issue. This provides stakeholders with visibility into the debugging workflow, helps avoid redundant efforts, and allows for a clearer assessment of the current resolution status. By including this element, the summary not only communicates what the problem is and why it occurred but also conveys how it has been addressed thus far. This approach provides a reproducible and efficient method for issue summarization, improving the accessibility of bug reports for developers, testers, and project managers, while simultaneously reducing the cognitive effort required to interpret raw technical logs.

4) Usability Testing

There are three types of testing conducted in this project including, system testing, integration testing and usability

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on January 29,2026 at 00:02:04 UTC from IEEE Xplore. Restrictions apply.

testing. The usability testing was conducted with 10 participants with experience in software development and testing. The results of the software testing and usability testing will be discussed in the next section.

III. Results and Discussion

A. Bug Severity Prediction

Based on table III, major has a lowest precision, recall and f1-score compared to minor and critical. This is due to imbalanced datasets. Total rows for each class: Minor is 24234, major is 3770 and critical is 3801. Although SMOTE technique is applied in this process to balance the classes, the model accuracy is improved but still did not achieve desired outcome. The synthetic samples from the SMOTE process may not capture the linguistic diversity of actual bug description, leading to poor generalization. Minor has 501 corrected predictions while critical has 493. These two models did very well when making prediction than major class as they have high precision, recall and F1-score.

TABLE III. Comparison Table of Precision, Recall and F1-Score

Class	Precision	Recall	F1-score
Critical	0.64	0.65	0.64
Major	0.52	0.48	0.50
Minor	0.60	0.64	0.62

To further enhance classification performance on imbalanced datasets, literature suggests combining oversampling techniques (e.g., SMOTE, ADASYN) with ensemble learning and boosting algorithms, for example SMOTE with AdaBoost [9, 10, 11, 12]. Oversampling helps by creating extra samples for the classes that don't have enough data (e.g., major), giving the model more balanced training examples. AdaBoost then strengthens this by paying extra attention to the cases the model often gets wrong, especially those from the smaller classes. Together, oversampling makes the smaller bug categories more visible, and AdaBoost makes sure the model learns from them, leading to better predictions for these less common bug types.

B. Bug Similarity

Table IV shows sample of the results based on the cosine similarity measures comparing the bug title together with the bug descriptions using the Kaggle dataset [3].

TABLE IV. Similarity Score for Each Pair of Duplicated Bugs

Bug title + Bug description	Score
expose a small simple UI for 'undo closed tab' bug 579361 added the backend changes to support 'undo closed tab' and implements a simple ctrl+shift+t shortcut key to undo the last closed tab. we should consider making a small ui to expose the feature too.	0.386
undo close tab (a way to quickly reopen a closed tab) It is easy to close a tab accidentally but no way to reopen it immediately (e.g. using a keyboard shortcut).	
script busy or stopped responding messages using 3.6 beta. I am using the beta version of firefox 3.6 under windows xp. i get the following pop-up message on my desktop: "a script on this page may be busy or it may have stopped responding. you can stop the script now or you can continue to see if the script will complete."script:file:///c:/program%20files/mozilla%20firefox%203.6%20beta%203/components/nsblocklistservice.js:648" sometimes this happens when i first open the browser. other times when i seek to restore a minimized browser that already is running. after i stop the script firefox opens (or restores) and i have no further problems with it.	0.68
warning: unresponsive script on startup - nsblocklistservice.js:648 - related to loading of plugins, sometimes i get the pop-up below when starting firefox 3.6. this window appears before the main firefox window appears. i did not see this with any previous 3.* version of firefox. warning: unresponsive script a script on this page may be busy or it may have stopped responding. you can stop the script now or you can continue to see if the script will complete.	
using personal certificate to connect an imaps server fails, leads to a failure to connect to an imaps server which requires a personal certificate to authorize the connection. steps to reproduce: 1. the server we have is a stunnel server providing the authentication 2. the configuration was working until 3.0.6 (included) since 3.1.1 the connection fails. I tried installing a fresh 3.1.1. (skipping the upgrade step) and recreated the accounts with the same result. going back to 3.0.6 solves the problem server log: ssl_accept: 140890c7: error:140890c7: ssl routines:ssl3_get_client_certificate:peer did not return a certificate	0.44
tls connection to imap server does not prompt for client certificate password, when connecting to an imap server using starttls and using a client certificate I am not prompted with the expected "please enter the master password for the software security device." so that the tls connection can be authenticated using the client certificate. steps to reproduce: 1. install a client certificate in the certificate manager 2. select starttls for the connection security in the server settings 3. connect to the server	

C. Bug Summarization

Fig. 6. AI-features experience

Based on Fig. 6, users demonstrated a high level of confidence in the bug summarization feature, which was perceived as the most beneficial AI-powered functionality within the system. The summaries were perceived as effective in facilitating a quick understanding of the bug context, especially for users without prior knowledge of the issue, thus reducing the necessity of reviewing the full report [13].

D. Usability Testing

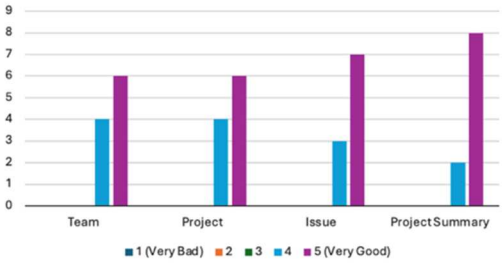

Fig. 7. Ease of use for each module

Based on Fig. 7, most modules including Team, Project, Issue, and Summary received scores between 4 (Good) and 5 (Very good) from the users. This indicates that most of the

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on January 29,2026 at 00:02:04 UTC from IEEE Xplore. Restrictions apply.

modules are perceived as very easy to use. These results support the usability goal of the proposed tool, demonstrating that users can interact with the tool efficiently and with minimal learning effort. Although the average score for the ease of use is between 4 (good) and 5(very good), some users also believed that there is confusion when completing their tasks in specific modules.

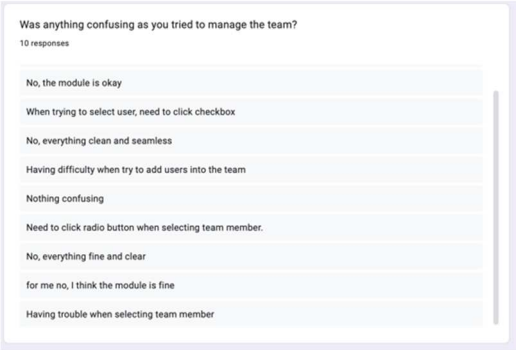

Fig. 8. Team module feedback

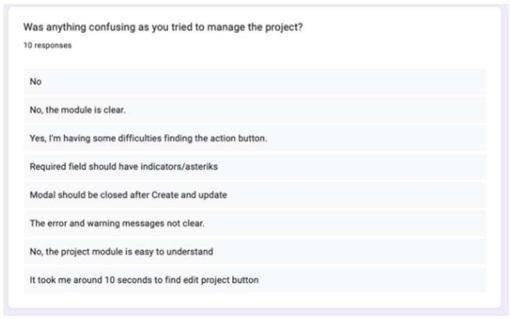

Fig. 9. Project module feedback

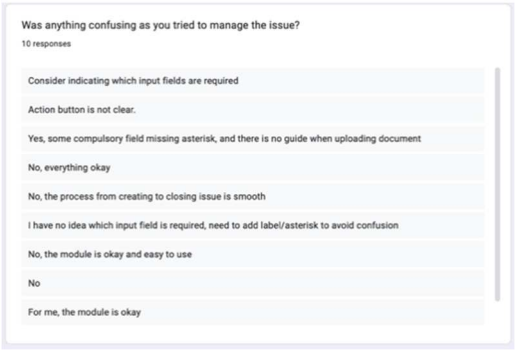

Fig. 10. Issue module feedback

Based on Fig. 8, majority of the users believed the team module did not confuse them. However, some of the users found certain parts of module is confusing while performing their tasks. The main issue occurred when users attempted to select a team member, as they were unsure whether to click the card or the radio button, leading to confusion during the selection process. In Fig. 9, majority of the users found nothing confusing in project module. Some of the users found certain parts of the project module confusing while performing specific tasks. Most of the issues were related to the user interface (UI), where the system failed to communicate effectively with the user such as bad "action" button placement, missing asterisk for required field, improper modal flow, and unclear error message. In issue module, some users found certain parts confusing when performing specific tasks. Users also found it difficult to look for "action" button and the form did not label each required field with asterisk. This causes users to spend a quite some time to look for action button as it is not at the desired area. Fig. 10 shows the feedback of issue module.

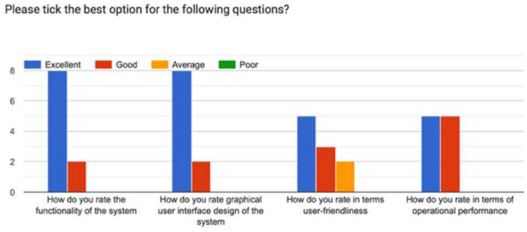

Fig. 11. Overall system assessment

Based on Fig. 11, the system received generally positive ratings across all usability dimensions. 8 out of 10 users rated the functionality of our proposed tool as 'Excellent', while the other two rated it as 'Good', suggesting that core features worked as expected and aligned well with user needs. The graphical user interface (GUI) was also well received, with 8 users selecting 'Excellent' and 2 selecting 'Good', indicating that the visual design and layout were clear and user-friendly. User-friendliness, however, received slightly more mixed feedback: 5 users selecting 'Excellent', 3 selecting 'Good', and 2 selecting 'Average'. The user who rated it as 'Average' expressed difficulty navigating certain modules, particularly when interacting with the team member selection interface and difficulty in finding the "action" button. This suggests that while the system is generally easy to use, some components may require improved guidance to support a wider range of user expectations. Operational performance was rated as 'Good' by 5 and 'Excellent' by 5, indicating that the system functioned reliably during use. However, the absence of full marks from all users may suggest minor delays or uncertainty about how fast the system should respond, especially during API interactions with the AI modules. Hence, while the usability profile is strong with most of the feedback between 'Good' and 'Excellent' the few non-optimal ratings reveal specific areas that could benefit from refinement. These include enhancing UI intuitiveness in certain modules and improving feedback mechanisms to make system responses more transparent and reassuring.

IV. CONCLUSION & FUTURE WORK

This paper addresses a highly relevant and practical problem in software engineering: automating bug tracking can significantly improve productivity, collaboration, and software quality, especially in large projects. The proposed AI-powered Software Bug Tracking Tool leverages on AI technologies such as machine learning to predict bug severity and classify bugs based on their patterns, enhancing the efficiency of bug tracking and resolution. The proposed system not only automates the categorization of bugs but also improves team communication and decision-making by

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on January 29,2026 at 00:02:04 UTC from IEEE Xplore. Restrictions apply.

providing intuitive tools for data visualization and task management. By involving the client throughout the design and development process, the system was tailored to meet the unique operational needs of TAHODC, laying the foundation for a more structured and effective approach to bug tracking. The limitation of project is the severity levels can be subjective and vary between organizations or projects. This may affect the consistency of predictions, especially in edge cases. Other than that, the UI/UX inconsistencies need to be eliminated and improved as some users experienced confusion in specific modules (e.g., team member selection, missing asterisk), indicating that parts of the interface require improvement for better usability. Finally, most modern tool nowadays able to integrate with another systems. The tool operates independently and may require additional integration work to be adopted into real-world development environments like Jira or GitHub issues.

For future work, the system should support customizable severity labels and criteria, allowing organizations to tailor the prediction model to their specific workflow. More advanced strategies could be explored to address class imbalance by including combining oversampling methods (e.g., SMOTE, ADASYN) with ensemble learning and boosting algorithms (e.g., SMOTE + AdaBoost). Such hybrid approaches, as stated in the literature, can enhance classification performance on imbalanced datasets and may help improve predictions for the major class. To address UI/UX inconsistencies, a revised acceptance test plan should be developed to improve feature intuitiveness (e.g., team member management) and standardize the placement of UI elements. Furthermore, the system could be extended to integrate with popular issue trackers such as Jira, GitHub Issues, and GitLab. This enhancement is expected to increase the practical impact and relevance of the proposed tool by enabling seamless bug reporting and analysis within existing development pipelines.

ACKNOWLEDGMENT

I would like to express my heartfelt gratitude to Baitulmal Sarawak for providing the funds for this project.

REFERENCES

[1] Lethbridge, T. C., & Laganière, R. (2004). Object-oriented software engineering: Practical software development using UML and Java. McGraw-Hill College.

[2] Krasniqi, R., & Do, H. (2023). A multi-model framework for semantically enhancing detection of quality-related bug report descriptions. *Empirical Software Engineering*, *28*(2), 42.

[3] Q. Liu, *Bugzilla Bug Reports*, Kaggle, Dataset. [Online]. Available: https://www.kaggle.com/datasets/qicongliu/bugzilla-bug-reports

[4] Chmielowski, L., & Kucharzak, M. (2021, June). Impact of software bug report preprocessing and vectorization on bug assignment accuracy. In *International Conference on Computer Recognition Systems* (pp. 153-162). Cham: Springer International Publishing.

[5] Behl, D., Handa, S., & Arora, A. (2014, February). A bug mining tool to identify and analyze security bugs using naive bayes and tf-idf. In *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)* (pp. 294-299). IEEE.

[6] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

[7] Jindal, R., Malhotra, R. & Jain, A. Prediction of defect severity by mining software project reports. *Int J Syst Assur Eng Manag* **8**, 334–351 (2017). https://doi-org.remotexs.unimas.my/10.1007/s13198-016-0438-y

[8] Kiran, P. S., Mahariba, A. J., Ramesh, D., & Sudheer, B. (2024, March). Software Defect Code Analyzer Using Cosine Similarity. In *International Conference on Recent Trends in Machine Learning, IOT, Smart Cities & Applications* (pp. 61-68). Singapore: Springer Nature Singapore.

[9] Benala, T. R., & Tantati, K. (2023). Efficiency of oversampling methods for enhancing software defect prediction by using imbalanced data. *Innovations in Systems and Software Engineering*, *19*(3), 247-263.

[10] Otoom, A. F., Al-Shdaifat, D., Hammad, M., & Abdallah, E. E. (2016). Severity prediction of software bugs. 2016 7th International Conference on Information and Communication Systems (ICICS). https://doi.org/10.1109/iacs.2016.7476092

[11] Hamouri, S. K., Shatnawi, R. A., AlZoubi, O., Migdady, A., & Yassein, M. B. (2023). Predicting bug severity using machine learning and ensemble learning techniques. 2023 14th International Conference on Information and Communication Systems (ICICS), 1-6. https://doi.org/10.1109/icics60529.2023.10330494

[12] Kurniawati, Y. E., & Prabowo, Y. D. (2022). Model optimisation of class imbalanced learning using ensemble classifier on over-sampling data. IAES International Journal of Artificial Intelligence, 11(1), 276. https://doi.org/10.11591/ijai.v11.i1.pp276-283

[13] Tarar, M. I. N., Ali, M., & Butt, W. H. (2019, October). Bug report summarization: A systematic literature review. In *Proceedings of the 11th International Conference on Education Technology and Computers* (pp. 257-261).

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on January 29,2026 at 00:02:04 UTC from IEEE Xplore. Restrictions apply.