“Silent Is Not Actually Silent”: An Investigation of Toxicity on Bug Report Discussion

Mia Mohammad Imran*
Missouri University of Science and Technology
Rolla, Missouri, USA
imranm@mst.edu

Jaydeb Sarker*
University of Nebraska Omaha
Omaha, Nebraska, USA
jsarker@unomaha.edu

Abstract

Toxicity in bug report discussions poses significant challenges to the collaborative dynamics of open-source software development. Bug reports are crucial for identifying and resolving defects, yet their inherently problem-focused nature and emotionally charged context make them susceptible to toxic interactions. This study explores toxicity in GitHub bug reports through a qualitative analysis of 203 bug threads, including 81 toxic ones. Our findings reveal that toxicity frequently arises from misaligned perceptions of bug severity and priority, unresolved frustrations with tools, and lapses in professional communication. These toxic interactions not only derail productive discussions but also reduce the likelihood of actionable outcomes, such as linking issues with pull requests. Our preliminary findings offer actionable recommendations to improve bug resolution by mitigating toxicity.

CCS Concepts

• **Software and its engineering** → **Open source teams**.

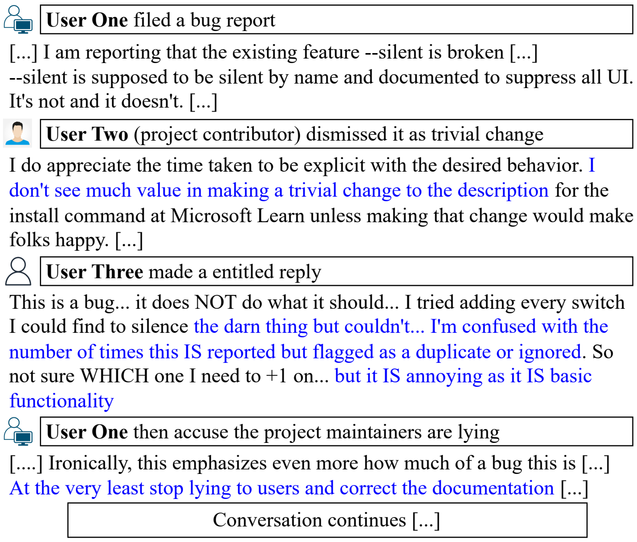
Keywords

toxicity, bug report, pull request discussions

ACM Reference Format:
Mia Mohammad Imran* and Jaydeb Sarker*. 2025. *“Silent Is Not Actually Silent”*: An Investigation of Toxicity on Bug Report Discussion. In *Companion Proceedings of the 33rd ACM Symposium on the Foundations of Software Engineering (FSE ’25), June 23–27, 2025, Trondheim, Norway.* ACM, New York, NY, USA, Article FSE029, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Bug reports serve as a means of communication between users and developers [29, 30]. They enable the identification and resolution of software defects, driving improvements in functionality and reliability [49]. Beyond their technical utility, bug reports help foster collaboration and build trust among community members [19, 34]. They provide a structured medium for discussing software issues, proposing solutions, assessing the severity of problems, and prioritizing tasks [8, 18, 43, 45, 49].



User One filed a bug report

[...] I am reporting that the existing feature --silent is broken [...]
--silent is supposed to be silent by name and documented to suppress all UI. It's not and it doesn't. [...]

User Two (project contributor) dismissed it as trivial change

I do appreciate the time taken to be explicit with the desired behavior. I don't see much value in making a trivial change to the description for the install command at Microsoft Learn unless making that change would make folks happy. [...]

User Three made a entitled reply

This is a bug... it does NOT do what it should... I tried adding every switch I could find to silence the darn thing but couldn't... I'm confused with the number of times this IS reported but flagged as a duplicate or ignored. So not sure WHICH one I need to +1 on... but it IS annoying as it IS basic functionality

User One then accuse the project maintainers are lying

[....] Ironically, this emphasizes even more how much of a bug this is [...]
At the very least stop lying to users and correct the documentation [...]

Conversation continues [...]

Figure 1: Toxic Interaction in Bug Report Discussion (blue marked text shows toxic phrase)

Bug reports require precise and professional communication, as they often contain technical descriptions and demand clear, constructive feedback to facilitate issue resolution. However, bug reports are not without challenges. Zimmermann *et* al. noted that *“there is a mismatch between what developers consider most helpful and what users provide”* in bug reports [49]. Prior research has revealed that bug reports are frequently incomplete and ambiguous, with their quality varying [9, 11, 47], which often leads to bug reports not being addressed. Ko *et* al. reported that users became dissatisfied when their bug reports were not addressed [29].

These frustrations can manifest in negative ways, including toxic interactions and other antisocial behaviors. Users may express impatience, dissatisfaction, or hostility when their concerns are dismissed or unresolved. Figure 1 illustrates a real-world bug report where the project maintainer dismissed a bug as a trivial change, sparking toxic comments. Toxicity in bug reports disrupts communication, hampers collaboration, and may discourage users from participating in or continuing to use the project. For example, in this bug report [4], the problem was not addressed, and the user was mocked by the owner. Subsequently, the user commented about switching to a different technology. In another thread [2], a user referenced market competition when a bug was not addressed (*“I really don’t mean to be total a*s about it but that’s not how you win market share.”*). These incidents show how toxicity in bug reports

[*] These authors contributed equally to this work.

FSE 2025, June 2025, Trondheim, Norway

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Data Sampling

Issue Thread	Count	ToxiCR (≥ 0.5)	LLaMA (≥ 0.7)
Total Thread	8,723	861	789
Selected Thread	203	186	165
Labeled as Toxic	81	80	59

can escalate, disrupt productive discourse, and ultimately derail the collaborative efforts necessary for resolving software issues.

While prior research has investigated toxicity in various broader Software Engineering contexts, including GitHub pull requests (PRs) and issues threads [15, 26, 35, 36, 39–42], bug reports remain an underexplored yet critical subset. Unlike other interactions, bug reports frequently highlight deficiencies in software, which can evoke emotions, such as frustration [23, 49], and lead to toxicity [7].

To address the gap in analyzing toxicity on bug reports, we have conducted a qualitative study on how toxicity manifests in bug reports and impacts the developers' bug resolution. To achieve this goal, we manually investigated 203 bug threads, including 81 with toxic interactions, which were selected using a stratified sampling strategy. After a qualitative exploration, we have found several notable findings:

- Toxicity arose when users perceived a disconnect between bug severity and maintainers' prioritization decisions.
- Toxic interactions hindered collaboration, with unresolved threads less likely to yield pull requests or fixes.
- Toxicity extended beyond interpersonal interactions and included negativity toward technology.

2 Methodology

In the following subsections, we have provided a brief description of our research methodology.

2.1 Dataset Creation and Labeling

2.1.1 Data Mining. We selected GitHub-based repositories for our study as it is the most popular open-source software project hosting platform [10, 14]. Primarily, we selected the repositories from GitHub for our analysis based on three criteria: i) at least 1000 stars to ensure sufficient community interest; ii) active in 2024 (up to September 5, 2024); and iii) active moderation was another key criterion. We defined it as evidence of maintenance and oversight, such as locking issues or the existence of a Code of Conduct included in the repository. After applying these criteria, we found 257 GitHub projects.

Further, we sampled from the repositories chosen based on the following criteria: i) all comments within threads must be in English to ensure consistency and accessibility; ii) created in 2024; iii) the issue/PR threads have at least 2 comments; and iv) each thread associated with a GitHub label related to bugs, such as `bug`, `defeat`, `crash`, `triage` or `fix` [24, 28]. Finally, we ended up with a total of 100 repositories and 8,723 threads with 91,929 comments.

2.1.2 Toxicity Classification. Since toxic conversations are rare in SE texts [35, 40], we employed automated tools to detect toxicity in the selected comments on our dataset. To improve the accuracy and reliability of our toxicity detection process, we utilized two toxicity detection methods. First, we selected ToxiCR [42], a state-of-the-art SE-specific toxicity detector. Second, we applied the LLaMA model to toxicity detection for our dataset. We provided the details in the following:

ToxiCR: A cutting-edge toxicity detection tool referred to as ToxiCR [42] has been specifically designed to detect toxic comments in code reviews, which also got attention from SE researchers [37]. Since this tool is publicly available with a 95.8% accuracy and 88.9% recall, we chose this toxicity detector for our analysis.

LLaMA Model: Since its inception, LLaMA [44] has become a popular LLM used for various affective analysis tasks, including toxicity detection [27, 31]. We chose the LLaMA model as it is an open-source model, unlike proprietary models like GPT-4. We utilized *LLaMA-3.1-70B* version. We asked the model to predict the likelihood of the conversation being toxic, given a thread on a scale of 0 to 1. We utilized the definitions of toxicity provided by Miller *et* al. [35] and incivility characteristics defined by Ferriera *et* al. [15] and Ehsani *et* al. [13]. The detailed prompt is included in the replication package [25].

Dataset Sampling: According to the authors of ToxiCR [42], a text would be marked as toxic if it scores ≥ 0.5. Hence, we set a threshold of ToxiCR as ≥ 0.5 and LLaMA toxicity detection as ≥ 0.7. We found 148 threads using these thresholds where both ToxiCR and LLaMA detected at least one toxic conversation. To expand our dataset size, we further sampled 55 threads where either ToxiCR or LLaMA has a toxicity score ≥ 0.99. Thus, we ended up with 203 threads, forming the basis of our toxicity analysis in bug reports. Table 1 shows the dataset selection for our analysis.

Manual Labeling: Toxicity is a complex phenomenon and varies according to different diverse demographic factors [32]. Moreover, the perception of toxicity differs in the SE domain from other social media platforms [35]. Therefore, two raters (authors of this paper) carefully reviewed prior literature on toxicity and other antisocial behaviors [16, 35, 40] in SE to reduce bias during manual labeling. They set five rounds of discussion sessions and collaboratively labeled the 203 threads [38]. They reviewed each thread, carefully read the comments, and marked the thread as toxic if at least one comment was labeled as toxic by two raters. At the end of this process, we found 81 toxic threads.

2.2 Qualitative Analysis Of Bug Threads

Inspired by recent works with toxicity and incivility analysis in the SE domain, we have conducted a qualitative analysis of selected toxic threads [15, 35]. We have manually analyzed which types of toxicity are more likely to occur in bug report discussions. We have found a total of 11 sub-categories of toxicity in our dataset. Table 2 shows our findings and subcategories of toxicity in GitHub bug discussions. We have also provided the mapping of subcategories in the second column of table 2 where two of them are not previously reported in any SE studies – '*Toxicity Towards Tools*' and '*Profane Technology Naming*'. Developers sometimes get frustrated with using some existing tools and use toxic comments toward that technology, which is referred to as the '*Toxicity Towards Tools*' category. Furthermore, we have found that the use of profane naming, including variable name method name, is an unprofessional practice, and we have defined that category as '*Profane Technology Naming*'. Further, two of our raters manually went through all the toxic threads and looked into the following factors:

- **Bug Resolved**: Whether the bug was resolved after the existence of toxic comments in the thread.
- **Authors of Toxicity**: We examined the authorship characteristics of toxic comments based on their membership within the project. GitHub provides the labels of the authors [17]. Authors were classified as internal participants if they were contributors, collaborators, members, and owners. Otherwise, they were categorized as external participants.
- **Toxic Reply**: We considered a toxic reply when a developer responded with toxic text after receiving a toxic comment from peers within the same thread.

3 Early Findings and Observations

In this section, we present the preliminary findings from our qualitative analysis of 81 toxic issue threads. Several noteworthy observations emerged, including those summarized in Table 2 and found in our analysis, which are discussed in detail below.

I. Subcategories of Toxicity: Table 2 shows 11 toxic categories that we found in our sample. Similar to prior studies on open source projects, we have found that vulgarity, insult, unprofessional, and entitlement are the most common categories encompassing 80% of the samples. In 13.58% cases, we observed toxicity directed toward tools, often characterized by negative or derogatory descriptions. While these comments were not aimed at individuals, they contribute to an unprofessional atmosphere and reveal deeper frustrations or dissatisfaction with the tools in question. For example, a user talked about missing docs in NextAuth by remarking - "Some examples would be awesome, docs or otherwise. The amount of hours wasted on nextauth is mindboggling. [...]" [5]. We observed that 2 of our threads contain tool's naming containing profanity (e.g., thef*ck). While these may not be intended to be toxic, they are inappropriate in a professional environment.

II. Bug Report Authors and External Participants Use More Toxicity: We observed that 41/81 (50.61%) toxic comments were authored by the same individual who created the bug report thread. This suggests that bug report authors are more likely to initiate toxic comments than other discussion participants. This behavior could stem from their heightened emotional investment in the issue, frustration over perceived neglect, or dissatisfaction with the responses they receive. Additionally, 56/81 (69.31%) first toxic comment authors were external participants. Though we worked with bug report threads, our results support the findings of a prior study [35]. An external participant may have many expectations and do not need to follow the Code of Conduct, which may trigger the use of more toxic comments than an internal participant. However, the internal participants authored 30.69% of toxic comments, which is not non-trivial, which indicates that active contributors could also be significant contributors to toxic discourse.

III. Bug Severity and Bug Priority: According to our observation, toxicity emerged when users perceived a bug as critical or a "showstopper," while maintainers either dismissed it or failed to address it promptly. This disconnect between users' perception of severity and maintainers' prioritization sparked frustration and led to toxic interactions. We found 17 threads where this happened. For example, in one bug report [3], a user commented - "this is such an annoying bug, this has existed for nearly half a decade; but the people working on this are more obsessed with useless things like server components.". Users occasionally questioned why certain bugs were not prioritized. For instance, one user [6], commented - "yes its frickin annoying. my whole project is on hold now."

IV. Bug Resolution Rate: Unsurprisingly, our analysis suggests that the presence of toxic comments negatively impacts the bug resolution rate. Among our sample, we found that a total of 36 bug threads (45%) were resolved. However, in threads containing vulgarity, insults, entitlement, and expressions of frustration, the bug resolution rate drops to approximately one-third, highlighting the detrimental effect of toxicity on resolution outcomes. This finding would help the project managers mitigate specific toxicity subcategories in the project discussions.

V. Absence of PRs from Toxic Threads: We observed that only 2 bug threads were PRs. This indicates that when a pull request addresses a bug, it is less likely to become toxic. One possible explanation is that pull requests are often tied to tangible progress and positive outcomes, such as bug fixes, which tend to elicit satisfaction and collaborative behavior rather than conflict [46, 48].

VI. Issue Bug Threads Linkage with PRs: GitHub bug reports that require *fixes* are often associated with pull requests [12, 33]. This indicates the bug reports are addressed in those issues. We found that 23/79 (29.11%) toxic bug report issues were linked with a PR that is lower than the percentage reported by Li *et* al. [33] (40.4%) and de Souza *et* al. [12] (35.7%). This discrepancy suggests that toxic bug threads are less likely to result in actionable fixes through PRs, indicating that they may be deprioritized or avoided altogether by the maintainers. Consequently, toxic threads are less likely to be addressed and more likely to remain unresolved, perpetuating frustration and dissatisfaction among users. For example, in one of the threads that was not linked to PRs, a collaborator provided a workaround solution, and it was promptly rejected by a user - "No. It's not. It's sloppiness and incompetence. And it's even less acceptable when it's actively used as an excuse not to revert a broken commit. [...]" [1].

VII. Subsequent Toxic Comments: In 24/81 (29.63%) cases, we observed subsequent toxic comments following the initial instance of toxicity. Only 6/24 cases, the actual problem faced in the issue, was resolved. This phenomenon was particularly evident in discussions where the initial comments contained insults or entitlement. Such interactions often created a cascading effect, escalating the negativity and derailing the conversation. For example, as illustrated in Figure 1, a dismissive or aggressive remark led to further toxic exchanges, making it difficult to refocus on the original issue.

VIII. Code of Conduct Enforcement: In 3 cases, the project maintainers enforced the Code of Conduct after toxicity occurred.

4 Implications and Future Directions

We have several insights and recommendations based on our study.

Transparent and Automated Bug Severity and Priority Management System: A robust automated system for managing bug severity and priority should ensure transparency and inclusivity by providing clear triage messages and consistent, visible labels such as "high-priority," "severity-high," or "needs clarification." Automated explanations for deprioritized or closed bugs, coupled with features like user voting on severity and real-time updates, can align

Table 2: Subcategories of Toxicity and Associated Factors (E = External participants, I = Internal participants)

Type	Mapping	Example	Count‡	Resolved	Toxic Reply
Vulgarity	[16, 35, 40]	Why is it so hard to install this thing? It has been 3 f**ing hours already!!!	19 (E: 16, I: 3)	31.6%	3
Insult	[35, 40]	It's really very annoying to debug [...] Is anybody alive here? Do you need reproduction? Or what?	18 (E: 14, I: 4)	38.9%	9
Unprofessional	[35, 40]	Darn it, now I can't reproduce after a restart of the window manager.	17 (E:10, I:7)	70.5%	0
Entitlement	[35]	IMO this is a disappointing outcome. No reliable guideline for anything but dbus/now.	14 (E:9, I:5)	35.5%	9
Toxicity Towards Tools	New	This means that Telegram Desktop [...] 5.0.1 and 5.0.2 contains shameful destructive bug	11 (E: 8, I: 3)	54.5%	1
Mocking	[16]	This isn't Windows compatible at all, are you joking?	4 (E: 4)	50%	2
Frustration	[16]	why the hell is no error message being reported, that is some real bug within podman. I still need to debug this.	3 (E: 1, I: 2)	33.33%	1
Profane Technology Naming	New	plugins=(git.... thefuck.....)	2 (E:2)	0%	0
Identity Attack	[40]	And then we don't have to pay Elon to message you...	1 (E:1)	100%	
Threat	[16, 40]	I think you're just trolling, so enjoy the ban and learn some manners.	1 (I:1)	100%	1
Arrogant	[35]	Whatever this function is supposed to achieve, it's very weird and messed up. Wouldn't it make more sense to fix this function like I suggested..	2 (I:2)	100%	0

‡ -*Since a text can fall into multiple sub-categories, the total count across all categories exceeds the sample size*

prioritization decisions with community needs while mitigating frustration. Recent research underscores the value of integrating social features with technical analysis to enhance bug-related processes [20–22]. A comprehensive system would leverage machine learning to predict severity, utilize sentiment analysis to assess urgency, and establish feedback loops for maintainers to explain decisions. Visual dashboards tracking bug resolution progress further promote transparency. These measures would foster trust, improve collaboration, and create a constructive environment for bug reporting while reducing the potential for toxicity.

Code of Conduct (CoC) for External Users: An outsider from a project tends to use for toxicity than a member of the project. For that reason, project maintainers should apply the CoC when creating the issues for external participants of the project.

Existing Tool Improvement and New Tool Development: While leveraging state-of-the-art technologies to identify toxic GitHub threads, we observed the presence of false positives, often caused by unique characteristics of bug reports, such as stack traces. Stack traces, crucial for diagnosing root causes, frequently include software engineering terminology that automated tools misinterpret as toxic. For instance, terms like '.ass' or 'nerd-font' in Stack Trace triggered false toxicity detections. Future tools should integrate domain-specific knowledge and context awareness to differentiate technical jargon from toxic language, ensuring more accurate detection without hindering productive discussions [22].

A large-scale empirical study: A potential research direction would be working on a large-scale study, which is necessary to provide more concrete and actionable recommendations for mitigating toxicity in bug report discussions. Such a study would enable the software engineering community to better understand the impact and associations of toxicity during the bug resolution process.

Limitation This study is limited to the analysis of 100 GitHub projects. Using the repository level and thread level criteria, we mitigate the external threat of selecting these repositories. We used ToxiCR [42] and LLaMA (*LLaMA-3.1:70B*) [44] for the selection of the toxic comments, which leads to construct validity. To mitigate this, we manually analyze whether the comments are toxic or not. However, we may miss some toxic comments that are not detected by each of the tools. Since we chose GitHub-based projects for analyzing the bug report toxicity, we may miss other software developers' platform where toxicity manifest differently. However, GitHub is the most popular and diverse platform for OSS, and thus, GitHub-based projects capture a wide range of toxic behaviors and diverse community interactions.

5 Conclusion and Future Work

Toxicity in bug reports poses significant challenges to collaboration and negatively impacts bug resolution rates. Through a qualitative analysis of 81 bug threads, we observed that external members are more prone to toxic comments, often driven by frustration over perceived mismatches between bug severity and maintainers' priorities. These insights emphasize the importance of addressing toxicity in project discussions to improve communication and efficiency. Our findings serve as a foundation for future work on developing automated systems to detect and manage toxicity in bug reports. Expanding this research through large-scale empirical studies and refining toxicity detection techniques remain essential goals for enhancing collaborative environments in software development.

References

[1] 2024. *containers/podman Issue 22190*. https://github.com/containers/podman/issues/22190

[2] 2024. *containers/podman Issue 22363*. https://github.com/containers/podman/issues/22363

[3] 2024. *facebook/react Issue 28584*. https://github.com/facebook/react/issues/28584

[4] 2024. *kovidgoyal/kitty Issue 7453*. https://github.com/kovidgoyal/kitty/issues/7453

[5] 2024. *nextauthjs/next-auth Issue 10633*. https://github.com/nextauthjs/next-auth/issues/10633

[6] 2024. *strapi/strapi Issue 19339*. https://github.com/strapi/strapi/issues/19339

[7] Magdaléna Balážiková. 2019. Real-life frustration from virtual worlds: The motivational potential of frustration. *Acta Ludologica* 2, 1 (2019), 56–68.

[8] Oscar Chaparro, Jing Lu, Fiorella Zampetti, Laura Moreno, Massimiliano Di Penta, Andrian Marcus, Gabriele Bavota, and Vincent Ng. 2017. Detecting Missing Information in Bug Descriptions. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, USA, 396–407.

[9] Vedant Chauhan, Chetan Arora, Hourieh Khalajzadeh, and John Grundy. 2024. How do software practitioners perceive human-centric defects? *Information and Software Technology* 176 (2024), 107549. https://doi.org/10.1016/j.infsof.2024.107549

[10] Kyle Daigle. 2023. Octoverse: The state of open source and rise of AI in 2023. https://github.blog/2023-11-08-the-state-of-open-source-and-ai/.

[11] Steven Davies and Marc Roper. 2014. What's in a bug report?. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 1–10.

[12] Cleidson RB de Souza, Emilie Ma, Jesse Wong, Dongwook Yoon, and Ivan Beschastnikh. 2024. Revealing Software Development Work Patterns with PR-Issue Graph Topologies. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 2402–2423.

[13] Ramtin Ehsani, Rezvaneh Rezapour, and Preetha Chatterjee. 2023. Exploring Moral Principles Exhibited in OSS: A Case Study on GitHub Heated Issues. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2092–2096.

[14] Hongbo Fang, Hemank Lamba, James Herbsleb, and Bogdan Vasilescu. 2022. “This Is Damn Slick!” Estimating the Impact of Tweets on Open Source Project Popularity and New Contributors. In *International Conference on Software Engineering (ICSE)*. ACM. https://doi.org/10.1145/3510003.3510121

[15] Isabella Ferreira, Bram Adams, and Jinghui Cheng. 2022. How heated is it? Understanding GitHub locked issues. In *Proceedings of the 19th International Conference on Mining Software Repositories*. 309–320.

[16] Isabella Ferreira, Jinghui Cheng, and Bram Adams. 2021. The" shut the f** k up" phenomenon: Characterizing incivility in open source code review discussions. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–35.

[17] GitHub. 2025. *Roles in an organization*. https://docs.github.com/en/organizations/managing-peoples-access-to-your-organization-with-roles/roles-in-an-organization

[18] Luiz Alberto Ferreira Gomes, Ricardo da Silva Torres, and Mario Lúcio Côrtes. 2019. Bug report severity level prediction in open source software: A survey and research opportunities. *Information and software technology* 115 (2019), 58–78.

[19] Mariam Guizani, Aileen Abril Castro-Guzman, Anita Sarma, and Igor Steinmacher. 2023. Rules of Engagement: Why and How Companies Participate in OSS. In *2023 IEEE/ACM 45th ICSE*. IEEE, 2617–2629.

[20] Yisi Han, Zhendong Wang, Yang Feng, Zhihong Zhao, and Yi Wang. 2024. Characterizing Developers' Linguistic Behaviors in Open Source Development across Their Social Statuses. *Proceedings of the ACM on Human-Computer Interaction* 8, CSCW1 (2024), 1–33.

[21] Zijie Huang, Zhiqing Shao, Guisheng Fan, Huiqun Yu, Kang Yang, and Ziyi Zhou. 2024. Bug report priority prediction using social and technical features. *Journal of Software: Evolution and Process* 36, 6 (2024), e2616.

[22] Mia Mohammad Imran, Preetha Chatterjee, and Kostadin Damevski. 2024. Shedding Light on Software Engineering-specific Metaphors and Idioms. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[23] Mia Mohammad Imran, Preetha Chatterjee, and Kostadin Damevski. 2024. Uncovering the causes of emotions in software developer communication using zero-shot llms. In *Proceedings of the IEEE/ACM 46th ICSE*. 1–13.

[24] Mia Mohammad Imran, Agnieszka Ciborowska, and Kostadin Damevski. 2021. Automatically selecting follow-up questions for deficient bug reports. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*.

[25] Mia Mohammad Imran and Jaydeb Sarker. 2025. Replication Package of Toxicity in Bug Reports. https://doi.org/10.5281/zenodo.15015619

[26] Mia Mohammad Imran, Robert Zita, Rebekah Copeland, Preetha Chatterjee, Rahat Rizvi Rahman, and Kostadin Damevski. 2025. Understanding and Predicting Derailment in Toxic Conversations on GitHub. *arXiv preprint arXiv:2503.02191* (2025).

[27] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674* (2023).

[28] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2021. Predicting issue types on GitHub. *Science of Computer Programming* 205 (2021), 102598.

[29] Amy J Ko and Parmit K Chilana. 2010. How power users help and hinder open bug reporting. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1665–1674.

[30] Amy J Ko and Parmit K Chilana. 2011. Design, discussion, and dissent in open bug reports. In *Proceedings of the 2011 iConference*. 106–113.

[31] Hyukhun Koh, Dohyung Kim, Minwoo Lee, and Kyomin Jung. 2024. Can llms recognize toxicity? a structured investigation framework and toxicity metric. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 6092–6114.

[32] Deepak Kumar, Patrick Gage Kelley, Sunny Consolvo, Joshua Mason, Elie Bursztein, Zakir Durumeric, Kurt Thomas, and Michael Bailey. 2021. Designing toxic content classification for a diversity of perspectives. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*. 299–318.

[33] Lisha Li, Zhilei Ren, Xiaochen Li, Weiqin Zou, and He Jiang. 2018. How are issue units linked? empirical study on the linking behavior in github. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 386–395.

[34] Jenny T Liang, Thomas Zimmermann, and Denae Ford. 2022. Understanding skills for OSS communities on GitHub. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 170–182.

[35] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian KaUstner. 2022. " Did you miss my comment or what?" understanding toxicity in open source discussions. In *Proceedings of the 44th International Conference on Software Engineering*. 710–722.

[36] Shyamal Mishra and Preetha Chatterjee. 2024. Exploring chatgpt for toxicity detection in github. In *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*. 6–10.

[37] Md Shamimur Rahman, Zadia Codabux, and Chanchal K Roy. 2024. Do Words Have Power? Understanding and Fostering Civility in Code Review Discussion. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 1632–1655.

[38] Johnny Saldaña. 2021. The coding manual for qualitative researchers. (2021).

[39] Jaydeb Sarker, Sayma Sultana, Steven R Wilson, and Amiangshu Bosu. 2023. ToxiSpanSE: An explainable toxicity detection in code review comments. In *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–12.

[40] Jaydeb Sarker, Asif Kamal Turzo, and Amiangshu Bosu. 2020. A benchmark study of the contemporary toxicity detectors on software engineering interactions. In *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 218–227.

[41] Jaydeb Sarker, Asif Kamal Turzo, and Amiangshu Bosu. 2025. The Landscape of Toxicity: An Empirical Investigation of Toxicity on GitHub. *Proceedings of the ACM on Software Engineering* FSE (2025), TBD.

[42] Jaydeb Sarker, Asif Kamal Turzo, Ming Dong, and Amiangshu Bosu. 2023. Automated identification of toxic code reviews using toxicr. *ACM Transactions on Software Engineering and Methodology* 32, 5 (2023), 1–32.

[43] Yang Song, Junayed Mahmud, Ying Zhou, Oscar Chaparro, Kevin Moran, Andrian Marcus, and Denys Poshyvanyk. 2022. Toward interactive bug reporting for (android app) end-users. In *Proceedings of the 30th ACM joint european software engineering conference and symposium on the foundations of software engineering*. 344–356.

[44] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[45] Wen-Yao Wang, Chen-Hao Wu, and Jie He. 2023. CLeBPI: Contrastive Learning for Bug Priority Inference. *Information and Software Technology* 164 (2023), 107302.

[46] Emily Winter, David Bowes, Steve Counsell, Tracy Hall, Sæmundur Haraldsson, Vesna Nowack, and John Woodward. 2022. How do developers really feel about bug fixing? directions for automatic program repair. *IEEE Transactions on Software Engineering* (2022).

[47] Nor Shahida Mohamad Yusop, John Grundy, and Rajesh Vasa. 2016. Reporting usability defects: do reporters report what software developers need?. In *Proceedings of the 20th international conference on evaluation and assessment in software engineering*. 1–10.

[48] Xunhui Zhang, Yue Yu, Georgios Gousios, and Ayushi Rastogi. 2022. Pull request decisions explained: An empirical overview. *IEEE Transactions on Software Engineering* 49, 2 (2022), 849–871.

[49] Thomas Zimmermann, Rahul Premraj, Nicolas Bettenburg, Sascha Just, Adrian Schroter, and Cathrin Weiss. 2010. What makes a good bug report? *IEEE Transactions on Software Engineering* 36, 5 (2010), 618–643.

Received 2025-01-16; accepted 2025-03-12