PeacemakerBot: A LLM-Powered Bot for Identifying and Reducing Signs of Incivility in GitHub Conversations

Antônio Gomes
Federal University of Ceará
Itapajé, Brazil
antoniocruz@alu.ufc.br

Eric Mesquita*
PUC-Rio
Rio de Janeiro, Brazil
jmesquita@inf.puc-rio.br

Emanuel Ávila
Federal University of Ceará
Fortaleza, Brazil
emanuel.pires@alu.ufc.br

Carlos Jefté
Federal University of Ceará
Itapajé, Brazil
carlosjefte@alu.ufc.br

Arthur Mesquita
Federal University of Ceará
Itapajé, Brazil
arthurwillame@alu.ufc.br

Lucas Sousa
Federal University of Ceará
Fortaleza, Brazil
lucassousa28@alu.ufc.br

Matheus Rabelo
Federal University of Ceará
Fortaleza, Brazil
matheusrabelo@alu.ufc.br

Mairieli Wessel
Radboud University
Nijmegen, The Netherlands
mairieli.wessel@ru.nl

Anderson Uchôa
Federal University of Ceará
Itapajé, Brazil
andersonuchoa@ufc.br

ABSTRACT

Context: Developers' interactions on collaborative software development platforms like GitHub are key to maintaining technical alignment and community engagement. However, uncivil behaviors such as disrespectful, sarcastic, or offensive comments can undermine these efforts, discouraging contributions and harming code quality. **Goal:** This study introduces PeacemakerBot, an automated moderation tool that detects and warns developers of incivility signs in GitHub conversations. **Method:** We leverage Large Language Models (LLMs) to analyze conversations, identify signals of incivility, and generate reformulation suggestions in real time. To evaluate it, we conducted a user study with six developers, followed by a survey based on the Technology Acceptance Model (TAM) to understand their perception of the tool's usefulness. **Results:** Our results suggest that PeacemakerBot successfully identifies multiple types of incivility and promotes more constructive conversations. The moderation feedback loop allows users to revise flagged comments, enhancing awareness and reducing harmful language over time. **Conclusion:** Our tool fills a key gap in OSS by providing AI-assisted moderation to enhance the social climate and inclusiveness of developer interactions. **Video link:** https://doi.org/10.5281/zenodo.15485535

KEYWORDS

incivility, conversations, github, llm, moderations, bot

1 Introduction

Open-source software (OSS) development enables global collaboration on software projects hosted on platforms like GitHub. Collaboration often happens through conversations on pull requests and issues [1, 21], where developers review code, share feedback, and discuss improvements. However, like any online interaction, these conversations can sometimes turn uncivil [8, 9], affecting the collaborative spirit on which OSS communities rely. In this context, conversations can range from helpful to rude or even uncivil, harming the sense of collaboration that open-source communities depend on [8, 11, 15]. For instance, comments such as *"Did you even bother testing this?"*, can discourage participation, harm team cohesion, and contribute to stress and burnout [8, 11, 15, 22].

Understanding uncivil behavior in software projects has drawn increasing attention in the software engineering community [7, 9, 15, 22, 26]. While these studies offer valuable insights into the nature of incivility in software development, there is still a lack of practical tools designed to address toxic behavior in conversations around software artifacts on collaborative development platforms [15, 20, 22, 25]. Existing moderation tools tend to be reactive, e.g., spam filters or bots that enforce codes of conduct only after offensive content is posted [15], and GitHub lacks built-in mechanisms to prevent incivility [9, 15] proactively. This gap highlights the need for actionable solutions to foster a more respectful and productive environment for contributors in open-source communities.

To contribute toward this goal, we introduce and evaluate *The PeacemakerBot*, an automated moderation tool designed to detect, monitor, and moderate uncivil behavior in pull requests and issues conversations on GitHub. The tool uses Large Language Models (LLMs) [24] to identify implicit and explicit signs of incivility. Once detected, PeacemakerBot automatically flags (potentially) uncivil comments in real-time, classifies them into specific types (e.g., mockery, impatience, insult), and provides up to three reformulated suggestions to encourage constructive alternatives. Developers can accept one of the suggestions or revise the comment manually. If the revised message remains uncivil, the bot offers additional suggestions and eventually shifts responsibility back to the developer, signaling that moderation is needed.

To assess its effectiveness and user acceptance, we conducted a synchronous user study with two teams of three software developers (6 developers in total). At the end of the study, we applied a survey based on the Technology Acceptance Model (TAM) [5] to evaluate their perceptions regarding the bot's usefulness, ease of use, and their intention to adopt. The study results show that developers found PeacemakerBot useful and easy to use, with most expressing a clear intention to adopt it in real development contexts.

*Also with Federal University of Ceará, Itapajé, Brazil.

2 Background and Related Work

Incivility & toxicity in OSS. Recent studies underscore the importance of addressing incivility and toxicity in OSS communities [6, 7, 9, 12, 15, 22, 26]. Miller et al. [15] identified different toxicity causes, such as arrogance and disrespect. Similarly, Ferreira et al. [8, 9] explored manifestations of incivility in Linux mailing lists and GitHub issues, focusing on how conversational cues such as tone, humor, and expressive style influence conversational dynamics. Ehsani et al. [6] analyzed issue threads and found that bitter frustration, impatience, and mocking were among the most prevalent forms of incivility, often leading to locked discussions due to heated exchanges. Imran et al. [12] highlighted that toxic language significantly impacts onboarding newcomers to OSS projects, emphasizing the need for proactive moderation strategies. We build PeacemakerBot on the taxonomy of incivility types proposed by Ehsani et al. [6] to inform our classification and intervention strategies within collaborative software development scenarios. The full description of each type of incivility is in our replication package.

Existing Moderation Tools. Recently, researchers have explored using NLP and LLMs for real-time toxicity detection [10, 16]. While GPT-based models perform well, they struggle with sarcasm and irony [16]. Transformer models like BERT outperform traditional classifiers, especially when sufficient context is given [10]. Tools like Climate Coach track toxicity retrospectively [19], while others use in-product reminders to encourage civility. For instance, Gerrit's Respectful Code Review Reminders aimed to reduce toxic comments, but Murphy-Hill et al. [17] found they had little impact, with toxicity sometimes increasing. While these tools rely on reactive or minimal interventions, they often fail to prevent conflict escalation or foster long-term behavioral change. In contrast, PeacemakerBot proactively mitigates conflict by guiding users to reformulate comments before harm escalates, educating them for future interactions. Unlike existing tools limited to post-hoc detection, PeacemakerBot integrates with GitHub's API to provide real-time, context-sensitive moderation, combining LLM automation with user-driven reformulation to enhance collaboration. Although the bot intervenes immediately after a comment is posted, its visible presence and timely feedback can foster anticipatory self-regulation as developers may preemptively reflect on tone and phrasing, knowing their comments will be analyzed. This helps build a more inclusive and constructive OSS environment.

3 PeacemakerBot in a Nutshell

PeacemakerBot integrates with pull requests and issues via webhooks to detect incivility in real time using LLMs. Users can adjust detection sensitivity and choose between two LLM providers (OpenAI and Groq API) to generate reformulated comment suggestions. The tool is released under the MIT License.

3.1 Architecture and Design Decisions

Figure 1 illustrates PeacemakerBot's three main components.

Front-End Interface. The PeacemakerBot front-end[1] is built with`React.js` and initialized with `Vite.js` to ensure fast build performance and efficiency. It supports user interaction via `GitHub OAuth`, enabling personalized access to moderation features. Upon

[1]https://github.com/reset-ufc/peacemaker-front-end

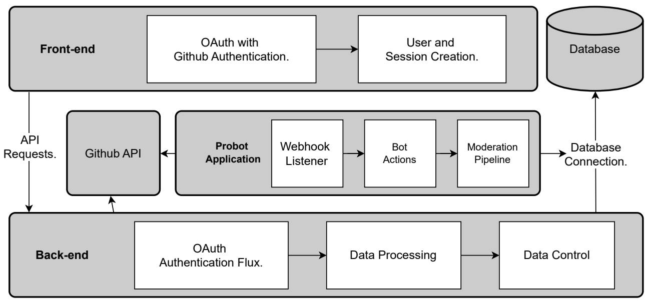


Figure 1: PeacemakerBot Architecture

authentication, the `User and Session Creation` component stores essential user metadata and manages sessions with `JWT` tokens. The interface communicates with the back-end through secure APIs to display inline moderation cues and context-aware reformulation suggestions, seamlessly integrating feedback into the developer workflow without disruption.

Back-End API. The back-end[2] is implemented using the `Nest.js` framework, chosen for its modular architecture and `TypeScript` support. The API exposes RESTful endpoints to support key operations, including storing flagged comments, retrieving reformulation suggestions, and logging of moderation actions taken by users. Data persistence is handled via a MongoDB database, chosen for its flexibility in managing semi-structured content such as user feedback and comment metadata. To ensure secure access, the system employs `OAuth2` authentication using `JSON Web Tokens` (JWT), which protects sensitive data and validates front-end requests during user interactions with the dashboard.

Probot Application. A critical component of PeacemakerBot is its integration with GitHub through a custom Probot application.[3] This application listens for webhook events, such as new comments on pull requests or issues, and forwards them to the bot action and the moderation pipeline. This event-driven mechanism ensures moderation is reactive, scalable, and tightly integrated with the GitHub collaboration ecosystem. Operating independently, the Probot app ensures that PeacemakerBot can react to real-time collaborative activity without requiring direct user interaction. This design allows decoupled event handling, enabling scalability and modular maintenance of GitHub-related logic. We explain the designed moderation pipeline as follows.

Moderation Pipeline with Human-in-the-Loop Feedback on Incivility Comments. We designed an automated moderation pipeline structured into three main steps: (1) *Incivility Detection*, (2) *Reformulation Generation*, and (3) *Reformulation Validation.*

In the *Incivility Detection* phase, PeacemakerBot leverages the Perspective API [13] to compute incivility scores for each user comment. Comments that exceed a user-defined incivility threshold (set to 75% by default) are flagged for intervention. During the *Reformulation Generation* phase, the flagged comment and its surrounding conversational context (up to $k = 3$ preceding comments) are passed to the `Llama-3.3-70B-Instruct` model by default, accessed via Groq Cloud API. The model performs two tasks: (i) it classifies the specific type(s) of incivility present in the original comment, and

[2]https://github.com/reset-ufc/peacemaker-api
[3]https://github.com/reset-ufc/peacemaker-bot

(ii) it generates up to three reformulated suggestions designed to preserve the comment's original intent while improving tone and civility. We set the LLM's temperature to 0.3 for suggestion generation, balancing creativity and reliability, and to 1.0 for classification, ensuring deterministic detection. Our replication package includes the prompt used to instruct the model, which contains classification and reformulation tasks along with few-shot examples.

Before these suggestions are presented to the user, each undergoes a *Reformulation Validation* process, in which the Perspective API is applied again to reassess the incivility of the AI-generated content. Only suggestions with incivility scores below the defined threshold are considered valid for presentation. The system performs up to three additional reformulation attempts if no valid suggestions are produced. However, if all validation rounds fail, the final three suggestions, regardless of their incivility scores, are presented to the user, ensuring continuity in the feedback process and preserving user agency. Once suggestions are presented, the human-in-the-loop mechanism enables users to take one of several actions (see Figure 2).

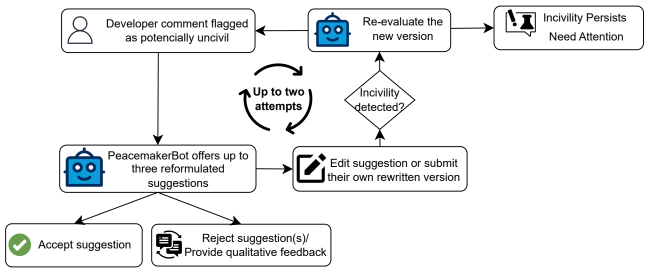
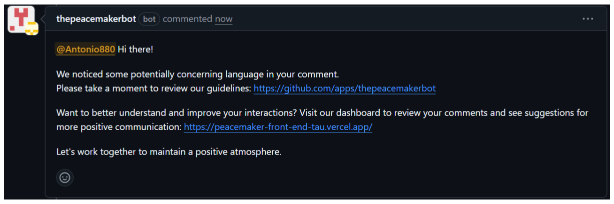

Developer comment flagged as potencially uncivil
Re-evaluate the new version
Incivility Persists Need Attention
Up to two attempts
Incivility detected?
PeacemakerBot offers up to three reformulated suggestions
Edit suggestion or submit their own rewritten version
Accept suggestion
Reject suggestion(s)/ Provide qualitative feedback

Figure 2: Human-in-the-Loop Moderation Workflow with Escalation to "Needs Attention"

Users can accept a suggestion, manually edit it, or submit their revised version. In cases where none of the suggestions are suitable, users may provide qualitative feedback about the suggestion (e.g., incorrect, inappropriate, not helpful, or other reasons) to help improve future LLM outputs. If a user decides to edit the comment manually, PeacemakerBot re-evaluates the new version using the same incivility analysis. If incivility is still detected, the system provides up to three new suggestions based on the updated input. If the comment remains uncivil after a second round of manual revision, it is marked as *"Needs Attention"*, and the system stops offering new reformulations, clearly indicating that the user has full responsibility for the content they choose to publish. This feedback loop lets users control the communication while benefiting from automated guidance. Therefore, by integrating LLM-generated support with human oversight and iterative feedback, we embrace a model of collaborative moderation in which AI acts as a supportive partner, enhancing rather than replacing human judgment.

3.2 Key Features and Interaction Flow

Initial Setup and Loading Project Data from GitHub. PeacemakerBot is installed as a GitHub App. Once added to a GitHub repository, it monitors real-time conversations, particularly in issues and pull requests. The integration is handled via GitHub's API, which allows the bot to dynamically load relevant project data such as discussion threads, user comments, and metadata (e.g., timestamps, authors, and thread type). This setup ensures that the system can analyze comments within their original context, reflecting the collaborative environment where the interactions occur. PeacemakerBot supports both public and private repositories (with the appropriate permissions). To initiate this process, users must: (1) select the repositories they want the bot to monitor; (2) click the Submit button to save these preferences. From that point on, PeacemakerBot actively monitors new comments as they appear. When it detects a potentially uncivil comment, the bot posts a contextual intervention directly in the conversation. This intervention encourages more respectful dialogue and includes a link redirecting the author of the comment to the system's dashboard (see Figure 3).


thepeacemakerbot bot commented now

@Antonio880 Hi there!

We noticed some potentially concerning language in your comment.
Please take a moment to review our guidelines: https://github.com/apps/thepeacemakerbot

Want to better understand and improve your interactions? Visit our dashboard to review your comments and see suggestions for more positive communication: https://peacemaker-front-end-tau.vercel.app/

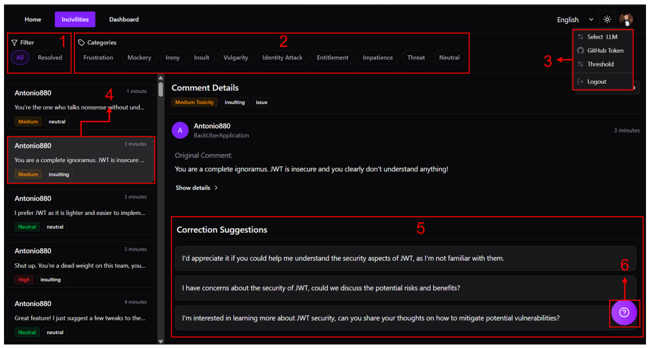
Let's work together to maintain a positive atmosphere.

Figure 3: Example of a PeacemakerBot intervention, including a link to the dashboard for more context

Configuring Detection Sensitivity and LLM Selection. PeacemakerBot allows users to define both the sensitivity of incivility detection and the choice of the LLM used for generating reformulations. As mentioned in the previous section, the users can configure a custom threshold to determine which comments should be flagged for deeper analysis (Fig. 4a, item ③). This score represents how likely the comment is to be perceived as incivility by a reader. Lower thresholds are ideal for educational or reflective environments, where even mild signs of incivility are relevant, while higher thresholds are better suited to stricter moderation settings that prioritize precision and aim to reduce false positives. Additionally, PeacemakerBot allows users to select from multiple LLM providers (OpenAI API and Groq API), including GPT-4 or LLaMA, to support a variety of usage scenarios and preferences. Users must provide their API keys for the selected provider, ensuring control over both data access and associated costs. We provided this dual configurability to make PeacemakerBot adaptable to different team cultures, project requirements, and resource constraints.

Incivility Detection & Suggestions. Figure 4 overviews the key features related to incivility detection and actionable suggestions.

Flagged Comments Panel. We provide a list of Flagged Comments along with their corresponding incivility category (Fig. 4a, item ④), enabling users to identify which comments have been marked as potentially uncivil quickly. ***Filter Controls.*** Allows users to filter the comment list by status (e.g., all or resolved) (Fig. 4a, item ①) and by up to ten types of incivility categories (Fig. 4a, item ②), supporting a more targeted and efficient analysis. ***LLM Correction Suggestions.*** Presents the original comment along with up to three reformulation suggestions generated by the LLM (Fig. 4a, item ⑤). These suggestions aim to retain the original intent while improving tone and promoting civil discourse. ***Suggestion Actions.*** Provides users with options to accept a suggestion, manually edit it, or reject all suggestions (Fig. 4b, item ⑧), ensuring full control over the final



Antonio880 · 1 minute · You're the one who talks nonsense without und... · Medium · neutral

Antonio880 · 3 minutes · You are a complete ignoramus. JWT is insecure ... · Medium · insulting

Antonio880 · 3 minutes · I prefer JWT as it is lighter and easier to implem... · Neutral · neutral

Antonio880 · 3 minutes · Shut up. You're a dead weight on this team, you... · High · insulting

Antonio880 · 4 minutes · Great feature! I just suggest a few tweaks to the... · Neutral · neutral

Antonio880 · BackUberApplication · 3 minutes

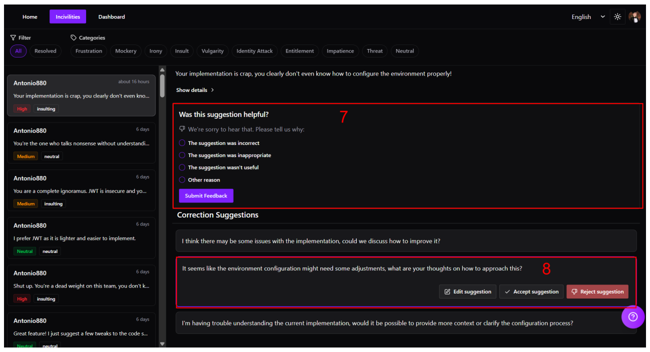
Original Comment:
You are a complete ignoramus. JWT is insecure and you clearly don't understand anything!

I'd appreciate it if you could help me understand the security aspects of JWT, as I'm not familiar with them.

I have concerns about the security of JWT, could we discuss the potential risks and benefits?

I'm interested in learning more about JWT security, can you share your thoughts on how to mitigate potential vulnerabilities?

(a) Detection panel with incivility categories and suggestions



Antonio880 · about 16 hours · Your implementation is crap, you clearly don't even kno... · High · insulting

Antonio880 · 6 days · You're the one who talks nonsense without understandi... · Medium · neutral

Antonio880 · 6 days · You are a complete ignoramus. JWT is insecure and yo... · Medium · insulting

Antonio880 · 6 days · I prefer JWT as it is lighter and easier to implement. · Neutral · neutral

Antonio880 · 6 days · Shut up. You're a dead weight on this team, you don't k... · High · insulting

Antonio880 · 6 days · Great feature! I just suggest a few tweaks to the code s... · Neutral · neutral

Your implementation is crap, you clearly don't even know how to configure the environment properly!

Was this suggestion helpful?
We're sorry to hear that. Please tell us why:
- ○ The suggestion was incorrect
- ○ The suggestion was inappropriate
- ○ The suggestion wasn't useful
- ○ Other reason

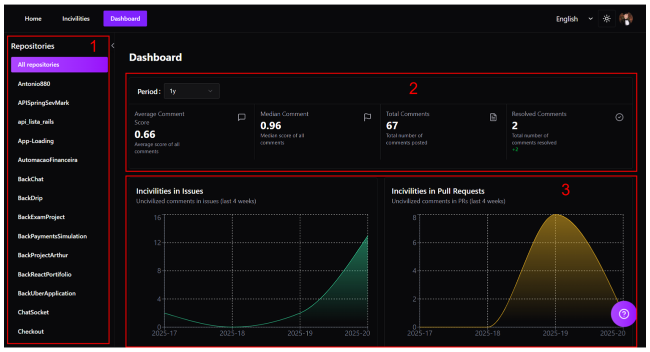
I think there may be some issues with the implementation, could we discuss how to improve it?

It seems like the environment configuration might need some adjustments, what are your thoughts on how to approach this?

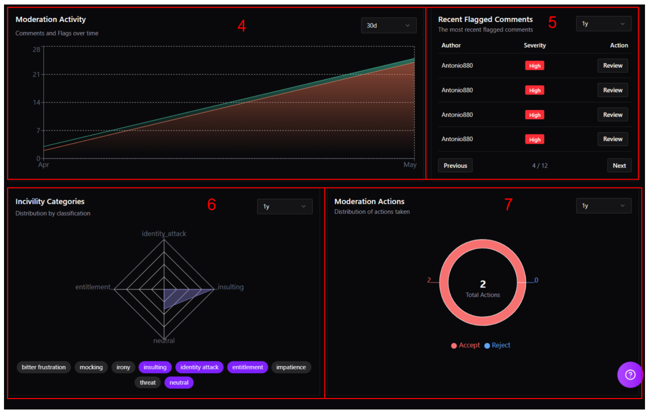
I'm having trouble understanding the current implementation, would it be possible to provide more context or clarify the configuration process?

(b) User actions to edit, accept, give feedback

Figure 4: Interface for incivility detection, reformulation suggestions, and user feedback actions.



(a) Top section of the conversation quality dashboard



Author	Severity	Action
Antonio880	High	Review
Antonio880	High	Review
Antonio880	High	Review
Antonio880	High	Review

Previous 4 / 12 Next

(b) Bottom section of conversation quality dashboard

Figure 5: PeacemakerBot Dashboard

content. ***Contextual Help.*** Includes a help button (Fig. 4a, item ⑥) that provides real-time guidance and clarification on how to set up and use the PeacemakerBot. ***Suggestion Feedback.*** Enables users to submit qualitative feedback on the generated suggestions (Fig. 4b, item ⑦), identifying issues such as inaccuracy, irrelevance, or lack of usefulness for further model refinement.

Conversation Quality Dashboard to Provide Signs of Incivility and Communication Insights. After analyzing project data, PeacemakerBot presents an interactive dashboard with key insights from the incivility detection process. It supports both high-level monitoring and detailed exploration of communication patterns. Figure 5 highlights its main components.

Repository Selector. Users can select specific repositories (Fig. 5a, item ①) to visualize either aggregated or project-level metrics. This feature enables contextualized analysis, supporting identifying incivility patterns within specific projects. ***Project Communication Summary.*** A consolidated overview (Fig. 5a, item ②) provides key communication metrics, including the average and median comment scores, the total number of analyzed comments, and the count of resolved comments. This allows users to monitor overall interaction quality. ***View on Incivility Trends by Interaction Type.*** Users can explore how incivility evolves across different collaboration modes using line charts segmented by issues and pull requests (Fig. 5a, item ③). This helps identify tone variation based on interaction type. ***View on Moderation Activity Timeline.*** Users can track the relationship between comment volume and flagged instances over time using an interactive timeline (Fig. 5b, item ④). This visualization supports monitoring moderation efforts and detecting trends in uncivil behavior. ***Recent Flagged Comments Panel.*** Users can review the most recently flagged comments in a dedicated panel (Fig. 5b, item ⑤), which displays details such as author, severity, and available moderation actions, enabling timely interventions. ***Incivility Classification Radar.*** Users can analyze the types of detected incivility through a radar chart (Fig. 5b, item ⑥). This component provides insight into the dominant types of incivility affecting project discourse. ***Moderation Decision Breakdown.*** Users can assess their engagement with the bot's suggestions via a pie chart (Fig. 5b, item ⑦) that categorizes decisions such as accepted, rejected, or edited. This enables reflection on how teams interact with automated recommendations.

4 Study Settings

Using the Goal-Question-Metric (GQM) template [2], we defined our goal as: **analyze** the PeacemakerBot tool; **for the purpose of** characterization; **with respect to** perceived usefulness, perceived ease of use, and intention to use; **from the viewpoint of** developers; **in the context of** collaborative software development projects on GitHub. Our research question is: *How do developers perceive the usefulness, ease of use, and their intention to use PeacemakerBot in collaborative software development?* Gathering developer feedback is essential to validate PeacemakerBot before pursuing broader deployment. To this end, we conducted a survey designed to obtain preliminary results on the developers' acceptance and perception of the usefulness of PeacemakerBot, and qualitative feedback on both its strengths and areas for improvement.

Survey Design. Our survey was based on the Technology Acceptance Model (TAM) [5], which is suitable for capturing users' acceptance of a given technology [29] and is frequently used to validate tools in the software engineering literature, e.g., [3, 23, 27, 28]. TAM guided the assessment of three key acceptance constructs: *perceived usefulness* (PU), *perceived ease of use* (PEOU), and *self-predicted future use* (SFU). PU reflects how much the technology is expected to enhance performance, PEOU refers to the effort needed to use it, and SFU indicates the user's anticipated intention to adopt it. Table 1 presents our survey items, organized according to the three TAM constructs. We used a 5-point Likert scale [14] to measure participants' agreement with each statement, ranging from *Strongly disagree* to *Strongly agree*, with a neutral option.

Table 1: Scale items for measuring PU, PEOU, and SFU

Perceived Usefulness (PU)
PU1. Using PeacemakerBot would help create a more civil and collaborative environment in my projects.
PU2. Using PeacemakerBot would improve communication among team members.
PU3. PeacemakerBot would help prevent conflicts and misunderstandings during software development.
PU4. PeacemakerBot would support maintaining a positive and respectful work environment.
PU5. I consider PeacemakerBot useful for encouraging good collaboration practices.
PU6. The interventions and suggestions provided by PeacemakerBot are useful for my daily collaborative work.
PU7. The visualizations offered by PeacemakerBot would help me reflect on the tone and quality of my communication over time.
PU8. The analyses and visualizations generated by PeacemakerBot help me monitor and improve my own communication practices.
PU9. I consider PeacemakerBot a useful tool for promoting more respectful and productive interactions in software projects.
Perceived Ease of Use (PEOU)
PEOU1. My interaction with PeacemakerBot is easy to understand.
PEOU2. The feedback and interventions provided by PeacemakerBot are clear and easy to follow.
PEOU3. Setting up PeacemakerBot in a collaborative project is easy and straightforward.
PEOU4. Connecting PeacemakerBot to a large language model via API is simple and intuitive.
PEOU5. Overall, I believe using PeacemakerBot would not require much effort.
Self-predicted Future Use (SFU)
SFU1. Assuming PeacemakerBot is available, I intend to use it in my future projects.
SFU2. I would recommend the use of PeacemakerBot to other developers.
SFU3. I believe PeacemakerBot would become a regular tool in my collaborative development practices.

Target survey population. We aimed to recruit developers experienced with collaborative platforms like GitHub, especially those active in pull requests, issues, or reviews, to assess PeacemakerBot's impact on civil and constructive communication. However, due to availability and recruitment context, the final sample included early-career developers. Participants were invited through direct contact and professional networks in software engineering. Of the 11 developers initially recruited, five did not attend the experiment sessions, resulting in a final sample of six participants who completed all tasks and provided feedback.

Table 2: Profile of Participants Grouped by Team

Team A								
ID	**Age**	**Education**	**Experience**	**Seniority**	**Frequency**	**Roles**	**Mod. Tool**	**GitHub Fam.**
P1	18-24yrs	HES	1-3yrs	Trainee	Daily	Dev, Scrum master	No	Low
P2	18-24yrs	HES	<1yr	Junior	Weekly	Dev	I am not sure	Low
P3	18-24ys	HES	1-3yrs	Junior	Weekly	Dev	I am not sure	High
Team B								
P4	18-24yrs	HES	<1yr	Junior	Weekly	Dev	No	High
P5	18-24yrs	HES	1-3yrs	Trainee	Weekly	Dev, Project Manager	No	Moderate
P6	18-24yrs	HES	1-3yrs	Trainee	Daily	Dev	I am not sure	Moderate

Note: HES = Higher Education Student; yrs = years

As shown in Table 2, all participants were students enrolled in higher education programs, with most reporting 1-3 years of development experience. Their roles were primarily as developers, with some also acting as Scrum Master or Project Manager. Two participants reported daily involvement in collaborative software development activities, while the others contributed weekly. Familiarity with moderation tools was low. Furthermore, most participants reported some exposure to pull requests or issue discussions. Although not fully aligned with the initial target population, this composition provided valuable early-stage insights into the usability and perceived utility of the bot in collaborative settings.

Tool usage and survey execution. We conducted two pilot tests (3 participants in each test) before the final experimental setup. These preliminary sessions were essential for refining the key features and the study procedures, validating the collaboration scenarios, and ensuring the clarity and adequacy of survey items. The participants involved in the pilot tests did not take part in the final setup. In the final setup, we instructed participants on the goal and procedures. We emphasized that participation was entirely voluntary and anonymous and that their responses would be used exclusively for academic research purposes.

To enable a consistent evaluation experience, participants interacted with a simulated GitHub project in which PeacemakerBot was integrated. In this project, developers were grouped into teams of three and assigned predefined roles to engage in six realistic collaboration scenarios, such as reviewing pull requests and discussing issues. Each scenario was designed to reflect common communication challenges and included at least one uncivil comment, allowing PeacemakerBot to intervene with suggestions to promote more civil and constructive communication. Sessions were conducted in a controlled lab setting ($\approx$ 30 minutes each), with researchers observing and noting interaction patterns and timing. Participants were encouraged to think aloud during the sessions to capture spontaneous reactions. After performing all scenarios, we asked participants to complete a survey that included the questions presented in Table 1. Additionally, we included two optional open-ended questions: *What improvement suggestions would you have? Did you miss anything?*, and *What positive aspects would you highlight about PeacemakerBot?*. The complete set of scenarios, instructions for role rotation, and environment setup are included in our replication package.

5 Evaluation Results

We present the results on PeacemakerBot's perceived usefulness, ease of use, and prediction of future use (see Figure 6).

Perceived Usefulness (PU). The results indicate an agreement among participants on the usefulness of PeacemakerBot in positively contributing to maintaining a respectful and cooperative environment during team interactions. As shown in Figure 6, 67% of participants strongly agreed with items PU3 and PU7, and none disagreed. These results suggest that the tool aligns with the developers' expectations to improve communication and team dynamics.

Perceived Ease of Use (PEOU). Most participants found PeacemakerBot easy to understand and use in day-to-day development tasks (PEOU1-PEOU4), with agreement rates reaching up to 83%. However, technical setup steps, such as connecting to the LLM (PEOU5 and PEOU6), showed lower agreement levels, suggesting that initial configuration remains a usability issue. These results point to a need for improved onboarding support.

Self-Predicted Future Use (SFU). Participants showed a strong intent to reuse and recommend PeacemakerBot. All agreed or strongly agreed with statements regarding future adoption (SFU1)

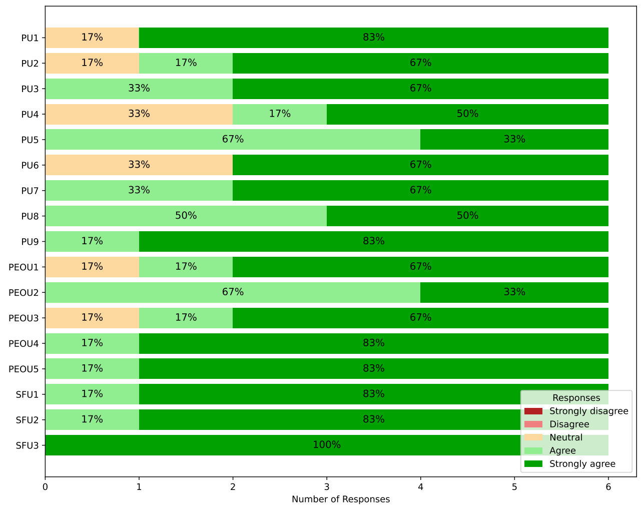


Figure 6: Distribution of Response to TAM Items

and recommendation (SFU2), the latter receiving 100% of agreement. This highlights the tool's perceived long-term value in collaborative settings. These results offer evidence of PeacemakerBot's perceived usefulness and usability. While some setup challenges were noted, they did not undermine the tool's overall positive reception. The high intent to reuse and recommend it highlights the relevance of its key features, real-time incivility detection, contextual feedback, and seamless workflow integration.

Open Feedback and Improvement Suggestions. Participants were also asked to provide open-ended feedback regarding the tool's strengths and possible improvements. The responses revealed both strong appreciation for the tool's usefulness and usability, as well as practical suggestions to enhance the user experience. Positive feedback emphasized the practicality and ease of use as stated by P6: *"A great application, easy to use, and very helpful in the daily routine of developers. It provides substantial support to the collaborative environment."*. Similarly, P3 noted that the tool is *"[...] simple, intuitive, and very useful for conversations in pull requests and issues."* These comments reflect the perceived alignment of the tool with developers' day-to-day collaborative needs.

A recurring theme emerged around the initial configuration process as a suggestion for possible improvement. P1 highlighted the need for clearer instructions and a more intuitive interface: *"Some instructions and the interface itself could be more intuitive."* P4 suggested shifting configuration responsibilities of the LLMs to the repository owner: *"The repository owner should configure the LLM token, so collaborators don't need to set anything up."* These observations suggest that the setup process may present a barrier to adoption for some users. Other suggestions focused on enhancing specific features. P2 proposed increasing the diversity of reformulation suggestions, P3 recommended improvements to the bot's visual identity, and P6 suggested adding multilingual support when the PeacemakerBot provides an intervention on GitHub. These observations reinforce the bot's value while indicating key areas for future improvement, particularly around onboarding, automation of configuration, and adaptability to different team contexts.

6 Tool Impacts and Limitations

PeacemakerBot fosters healthier OSS communication by providing a real-time human-in-the-loop moderation workflow that prevents misunderstandings and conflicts. Despite promising results, we identified limitations that offer opportunities for improvement:

GitHub OAuth and Token Setup. The OAuth token generated via GitHub's API does not provide sufficient permissions to edit comments. As a result, users must manually generate and configure a classic personal access token with the appropriate permissions. Combined with the need to configure both GitHub and LLM API keys, this setup adds friction, especially for non-technical users. While a setup guide is available, future versions should aim to simplify this process to enhance adoption and usability.

LLM Context Handling and Cost. While contextual inputs enhance moderation quality and reduce hallucinations, they significantly increase token usage and API costs. This trade-off between contextual fidelity and computational efficiency remains a challenge for scalable deployment. Future iterations could explore techniques such as selective context injection, caching strategies, or lightweight model variants may help mitigate cost without severely impacting performance [18]. ***Sustainability Concerns.*** The tool's use of multiple LLM inference cycles per comment, including retries and validation, can lead to substantial energy and API consumption, particularly in high-traffic repositories. To help mitigate this, PeacemakerBot employs the Perspective API as a pre-filtering heuristic to identify potentially uncivil comments before invoking more costly LLM-based moderation. However, additional strategies such as model size reduction [18] or enforcing rate-limiting mechanisms to control inference load remain essential for ensuring scalability and minimizing environmental impact.

7 Conclusion and Future Work

We introduced PeacemakerBot, an LLM-powered moderation assistant designed to foster respectful communication in GitHub conversations. Unlike reactive tools that merely flag comments, PeacemakerBot offers real-time, actionable reformulation suggestions, helping educate users and reduce future incivility. Our evaluation showed strong user acceptance, especially regarding its usefulness and impact on promoting civil discourse. For future work, we plan to improve the system's architecture by refining how the bot connects to its back-end services and database, aiming for greater scalability and stability. We also intend to simplify the configuration experience by redesigning the user interface for setting up LLM and GitHub tokens, ideally supporting automatic detection of permissions and centralized token management by repository maintainers. Lastly, we will enhance error handling for API usage, particularly in scenarios of rate limits and token exhaustion from the LLM provider, providing more robust feedback to users.

ARTIFACT AVAILABILITY

Our replication package is publicly available at [4].

ACKNOWLEDGMENTS

This study was financed by PIBITI-FUNCAP, PREX-UFC, and FUNCAP (BP6-00241-00276.01.00/25).

REFERENCES

[1] Caio Barbosa, Anderson Uchôa, Daniel Coutinho, Wesley KG Assunçao, Anderson Oliveira, Alessandro Garcia, Baldoino Fonseca, Matheus Rabelo, José Eric Coelho, Eryka Carvalho, et al. 2023. Beyond the Code: Investigating the Effects of Pull Request Conversations on Design Decay. In *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–12.

[2] Victor R Basili-Gianluigi Caldiera and H Dieter Rombach. 1994. Goal question metric paradigm. *Encyclopedia of software engineering* 1, 528-532 (1994), 6.

[3] Hui Hui Chen, Ming Che Lee, Yun Lin Wu, Jing Yao Qiu, Cheng He Lin, Hong Yong Tang, and Ching Hui Chen. 2012. An analysis of moodle in engineering education: The TAM perspective. In *International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE, H1C–1.

[4] Antônio Cruz Gomes, Eric Mesquita, Emanuel Ávila, Carlos Jefté, Arthur Mesquita, Lucas Sousa, Matheus Rabelo, Mairieli Wessel, and Anderson Uchôa. 2025. *Replication package for the paper: "PeacemakerBot: A LLM-Powered Bot for Identifying and Reducing Signs of Incivility in GitHub Conversations"*. doi:10.5281/zenodo.15485535

[5] Fred D Davis. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly* (1989), 319–340.

[6] Ramtin Ehsani, Mia Mohammad Imran, Robert Zita, Kostadin Damevski, and Preetha Chatterjee. 2024. Incivility in Open Source Projects: A Comprehensive Annotated Dataset of Locked GitHub Issue Threads. In *21st MSR*. IEEE, 1–5.

[7] 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.

[8] 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.

[9] 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.

[10] Isabella Ferreira, Jinghui Cheng, and Bram Adams. 2024. Incivility detection in open source code review and issue discussions. In *Proceedings of the 21st International Conference on Mining Software Repositories (MSR)*. IEEE, 1–11. doi:10.1145/3620309.3620320

[11] Daviti Gachechiladze, Filippo Lanubile, Nicole Novielli, and Alexander Serebrenik. 2017. Anger and its direction in collaborative software development. In *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER)*. IEEE, 11–14.

[12] 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).

[13] Jigsaw and Google. 2017. Perspective API. https://perspectiveapi.com/. accessed July 19, 2025.

[14] R. Likert. 1932. *A Technique for the Measurement of Attitudes*. Number Nº 136-165 in A Technique for the Measurement of Attitudes. Archives of Psychology.

[15] 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.

[16] 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.

[17] Emerson Murphy-Hill, Jill Dicker, Delphine Carlson, Marian Harbach, Ambar Murillo, and Tao Zhou. 2024. Did Gerrit's Respectful Code Review Reminders Reduce Comment Toxicity? In *Equity, Diversity, and Inclusion in Software Engineering: Best Practices and Insights*. Apress Berkeley, CA, 309–321.

[18] Mario Patrício, Silas Eufrásio, Anderson Uchôa, Lincoln S. Rocha, Daniel Coutinho, Juliana Alves Pereira, Matheus Paixão, and Alessandro Garcia. 2025. Civility Not Found? Evaluating the Effectiveness of Small Language Models in Detecting Incivility in GitHub Conversations. In *Proceedings of the XXXIX Brazilian Symposium on Software Engineering (SBES)*. SBC.

[19] Huilian Sophie Qiu, Anna Lieb, Jennifer Chou, Megan Carneal, Jasmine Mok, Emily Amspoker, Bogdan Vasilescu, and Laura Dabbish. 2023. Climate coach: A dashboard for open-source maintainers to overview community dynamics. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–18.

[20] Huilian Sophie Qiu, Bogdan Vasilescu, Christian Kästner, Carolyn Egelman, Ciera Jaspan, and Emerson Murphy-Hill. 2022. Detecting interpersonal conflict in issues and code review: cross pollinating open-and closed-source approaches. In *Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society*. 41–55.

[21] Mohammad Masudur Rahman and Chanchal K Roy. 2014. An insight into the pull requests of github. In *Proceedings of the 11th working conference on mining software repositories*. 364–367.

[22] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. 2020. Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 57–60.

[23] Jonan Richards and Mairieli Wessel. 2024. What You Need is what You Get: Theory of Mind for an LLM-Based Code Understanding Assistant. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 666–671. doi:10.1109/ICSME58944.2024.00070

[24] Steven I Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D Weisz. 2023. The programmer's assistant: Conversational interaction with a large language model for software development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*. 491–514.

[25] 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.

[26] 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.

[27] Marcos Silva, Maykon Nunes, Carla Bezerra, Anderson Uchôa, and Mairieli Wessel. 2023. CIRef: A Tool for Visualizing the Historical Data of Software Refactorings in Java Projects. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. 174–179.

[28] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. 2016. Overcoming open source project entry barriers with a portal for newcomers. In *38th International Conference on Software Engineering*. 273–284.

[29] Mark Turner, Barbara Kitchenham, Pearl Brereton, Stuart Charters, and David Budgen. 2010. Does the technology acceptance model predict actual use? A systematic literature review. *Inf. Softw. Technol.* 52, 5 (2010), 463–479.