Topics, Trends, and Sentiments in Software Testing: An Analysis of Developers' Engagement on Stack Overflow

Anthony Wambua Wambua
Department of Computer Science,
Daystar University,
Athi River, Kenya
Email: awambua [AT] daystar.ac.ke

Abstract----**This study investigated software testing discussions on Stack Overflow from 2020 to 2024 to uncover key trends, topics, and developer sentiments. 14 key topics, including unit testing, machine learning testing, mobile testing (especially Flutter), and Docker testing were identified. The study revealed a decline in developer engagement, as the number of posts answered and with accepted answers decreased, particularly after 2022. Sentiment analysis showed a predominance of negative sentiments across most topics, especially in mobile and machine learning testing. While some topics like machine learning testing initially saw positive sentiment, this shifted toward frustration as the years progressed. These findings suggest that the rise of AI-based tools, such as ChatGPT, has affected the way developers engage with traditional forums like Stack Overflow. The decline in engagement and the prevalence of negative sentiments highlight the challenges developers face in software testing. This research points to the need for further investigation into how AI tools influence developer behavior and their reliance on peer support platforms. Additionally, it suggests exploring how sentiment analysis can be integrated into software testing tools to better address developer frustrations and improve support for testing emerging technologies. The study provides insights that could guide the development of more effective tools and frameworks to enhance the software testing process.**

Keywords-Software Testing, Stack Overflow, Topic Modeling, Sentiment Analysis, Developer Engagement, Machine Learning Testing, ChatGPT

I. INTRODUCTION

Software testing involves applying inputs to a system under test (SUT) to isolate factors affecting its correctness [1]. This ensures that the software does not have bugs that can cause critical failures once deployed. Software failures cause disruptions, user mistrust [2] and have been estimated to cost the US economy about $2.08 trillion annually [3]. For this reason, software developers have intensified software testing. The testing cost is estimated to cost at least 50% of the overall cost of a project [4, 5]. Software testing is complex, requiring curated skills, training [6], familiarity with specific tools and frameworks [7], and the need to tailor the testing to the specifics of the SUT. As such, as developers learn to test or test their code, they are bound to encounter errors and issues that lead them to post on or read from open developer forums such as Stack Overflow, where they can get assistance from other developers [8-10].

Over the years, Stack Overflow has become the go-to troubleshooting site for developers, amassing much data [11] from developers regarding software engineering, specifically software testing, and how developers engage with software testing matters. Careful mining and analysis of the data can help the software engineering community draw insights regarding the discussion of developers in these forums. Consequently, this study seeks to analyze software testing discussions of developers on Stack Overflow. It explores trends, sentiments, topics of discussion, and developers' engagement over a five-year period.

Specifically, the study addresses the following research questions:

RQ1: What key topics and trends emerge in software testing discussions on Stack Overflow using topic modeling techniques? The motivation is to establish the topics and themes that form the basis for software testing discussion, as this can provide insights into the issues developers are grappling with regarding software testing.

RQ3: How have software testing topics on Stack Overflow evolved? Understanding how topics related to software testing have evolved can provide valuable insights into industry trends. Given the dynamic nature of the software industry, we are motivated to examine whether discussions on software testing evolve alongside changes in tools, methods, frameworks, and programming languages.

RQ2: How do developers engage in software testing discussions? Stack Overflow is a platform that developers rely on for solutions. We are motivated to explore the number of questions posted, how many questions receive answers

versus those that remain unanswered, and how this has evolved. Additionally, we are curious about the potential impact of tools like ChatGPT on developer engagement in Question-Answer (QA) forums.

RQ4: What sentiments do developers express in software testing discussions, and how do they relate to different topics? The objective is to understand developers' sentiments toward various topics in software testing discussions and how these sentiments evolve. This insight can help improve support for developers as they test their products.

This study's contributions are summarized as follows:

- We apply topic modeling to uncover significant topics in software testing discussions on Stack Overflow.
- We analyze how software testing topics change over time, reflecting shifts in industry tools and practices.
- We assess developer participation in testing discussions by examining answered, unanswered, and accepted questions.
- We evaluate developer sentiments on software testing topics and how they evolve.

The rest of the paper is structured as follows: The Background section explores key concepts related to software testing, developer forums, and natural language processing, providing context for the study. The Method section outlines our approach to addressing the research questions. This is followed by the Results section, where we present our findings. In the Discussion section, we analyze and interpret these findings. Finally, the Conclusion section summarizes the study, offers recommendations, and outlines directions for future research.

II. BACKGROUND

A. Software testing

The significance of software testing in the software development lifecycle cannot be emphasized as a means to ensure software quality and reliability [2]. As a result, researchers and the industry have concentrated on developing tools, frameworks, and strategies to enhance and streamline the software testing process among software developers [12]. Owing to the time and cost of software testing, there has been a shift from manual testing to automated testing, where computer tools and frameworks are used to carry out software testing automatically [13].

Various types of software testing are performed at different stages of development, each serving distinct objectives. Unit testing, carried out by developers during the development phase, focuses on testing individual components of the software to ensure they function as expected [14]. Tools like JUnit, NUnit, and Pytest are commonly used. Once the components are integrated, integration testing is performed to validate the interactions between different modules. This testing is done by developers or testers using frameworks such as JUnit [15] and SoapUI [16].

After integration, testers conduct system testing to validate the entire system's functionality before releasing the software [17]. Tools such as Selenium and LoadRunner are commonly employed for this phase. Testers also conduct regression testing to ensure that new changes or bug fixes do not introduce issues in existing functionality [18]. This is typically done using tools like Selenium and QTP.

In addition to these core testing types, other important testing practices include security testing to identify vulnerabilities, performance testing to evaluate the software's behaviour under stress, acceptance testing to confirm that the software meets business requirements, and usability testing and compatibility testing to ensure that the software is user-friendly and functions across different platforms and environments [12].

Other important and recent events in software testing include using large language models (LLMs) for software testing. Researchers have begun to explore their applicability in areas such as test case generation [19], program repair [17] and defect detection [20]. Further recent developments such as DevOps, which advocates for faster development and more efficient collaboration between development and operations, have led to continuous integration (CI) and continuous delivery (CD), where new code is integrated into the main code base often and frequently [21]. As such, new testing approaches, tools and frameworks are bound to be used.

While research has focused on developing new tools, frameworks, approaches, and strategies to optimize testing, it is equally important to understand how developers carry out testing, the tools they use, and their sentiments toward software testing. Additionally, it is crucial to explore how software testing has evolved given changes happening in the software engineering field and how developers are engaged in this subject. Studying developer QA forums can yield these insights. Such understanding is crucial as it sheds light on a critical undertaking of the software development lifecycle.

B. Stack overflow

Stack Overflow (SO) is one of the many question-answer (QA) forums used by developers to post and get answers to their software engineering-related questions. The website has been around since 2008 and has risen in popularity with over 10 million contributors and having amassed over 23 million questions and 35 million answers by 2023 [22, 23].

On the site, a user (questioner) can post a question asking for help. Other users (answerers) can respond to the question. A single question can have many answers. When a user is satisfied with any of the answers, they can mark such an answer as the accepted answer for that question. As other users visit the question, the question's view increases, meaning a question with many views has attracted developers' interest. Further, users can upvote or downvote any question or answer that they find helpful or unhelpful, respectively. Therefore, a post's (question or answer) score can communicate how the community feels about the post's

importance. As a user creates a question, they provide a title, body and tags for ease of discovery. Words used in a post's tags, title and body can be used to discover posts related to specific topics, such as software testing.

C. Topic modelling

The rapid advancement of technology has led to the generation of large-scale datasets across a wide range of platforms by diverse contributors. This exponential growth in data presents significant challenges in extracting meaningful insights. To address this, automated methods, such as topic modeling, are employed to uncover the underlying themes and topics within these voluminous corpora. Topic modeling is a natural language processing (NLP) technique that automatically discovers topics by clustering words in a document [24, 25]. The technique uses unsupervised machine language approach to group together words that are associated with a certain topic [26].

A popular topic modeling technique is Latent Dirichlet Allocation (LDA) which assumes each document consists of a mixture of topics- bag of words (BOW), and each topic a mixture of words. It therefore uses probabilistic methods to assign words to topics in an interactive manner. LDA can be supervised or unsupervised and requires no previous training [27]. For LDA to output a list of topics and the associated topic terms, it must be supplied with the number of topics (N). The number of topics can be manually assigned or automatically computer by an algorithm, this number is critical as it determines the success of the topic modelling exercise [22, 25]. One of the common approaches to determining the optimal number of topics is Perplexity which refers to how well a model fits with the probability distribution [28], it can be thought of a measure of surprise or certainty in predicting the next word in a sequence. Normally, an algorithm receives a range of topics and computes the perplexity for each value in the range. Lower values of perplexity indicate an optimal number of topics. This approach has been used in numerous studies such as [29-31]

D. Sentiment Analysis

Sentiment analysis is an NLP technique that analyzes people's thoughts, feelings and opinions regarding certain subjects such as services or products [32]. Unlike other text processing approaches, sentiment analysis focuses on classifying user opinions and attitudes into either positive, negative or neutral [33], allowing the processing of huge datasets without having to reading each user's views. For this reason, sentiment analysis has gained popularity as it allows automatic monitoring of user views in fields like politics, business and software engineering. In software engineering, sentiment analysis can be used to understand what users feel about certain application by performing sentiment analysis of app reviews on Google play. In this study, sentiment analysis will be applied to understand what developers feel about the topics they discuss concerning software testing on Stack Overflow.

III. METHODOLOGY

To analyze discussions on SO and identify topics, trends, sentiments, and developers' engagement regarding software testing, we adopted a mixed methods research approach. To answer RQ1 and RQ3, we conducted LDA topic modelling to identify topics' terms and then manually inferred the topics. To answer RQ2, we used descriptive statistics, while for RQ4, we conducted sentiment analysis. Fig. 1 depicts the research methodology, which is further explained in the following subsections.

A. Experimental setup

The study used a Dell Precision 5550 desktop computer running on Windows 11, equipped with 64GB RAM, a Core i7 vPRO processor with 12 CPUs, a 1TB SSD, and both Intel and NVIDIA GPUs. The GPUs were very handy in accelerating sentiment analysis, which is a resource-intensive undertaking. A Python program was created utilizing the rich Python library collection for the different operations discussed in the procedure subsection.

B. Data collection

Data from SO was collected on 1st February 2025. Our initial attempt to extract data directly from SO into our Python program environment through the SO API did not work owing to API quota restrictions despite registering as required. We used the data explorer, which utilizes query to fetch data and return a CSV file containing the requested data. Since we needed data for five years, our query kept timing out. To fix this, we extracted posts year by year from 2020-2024, all with the word "test" as a tag within the post body or title. The data mining exercise, depicted in Fig. 1, yielded 120,909 posts, which were then used in our analysis to answer the research questions.

C. Procedure

Data preprocessing: After data was downloaded from SO, it was merged into a single CSV file containing 120,909 posts. The data contained fields such as title, body, tags, accepted answer ID, score, view count, and question ID. The CSV file was loaded in a Python program to preprocess the data. HTML tags and special characters were removed, and the text was converted to lowercase using the *BeautifulSoup* library. Given the massive amount of data, the preprocessing utilized parallel execution through the *concurrent.futures* module available in Python.

Descriptive statistics: To answer RQ2 regarding developers' engagement in software testing, descriptive statistics of the downloaded data were computed. This helps establish the questions posted each year, the number of answered and unanswered questions, and those with accepted answers.

LDA topic modeling: To answer RQ1 and RQ3, we started by determining the optimal number of topics. This was done by computing the perplexity of topics from 1 to 15; lower

perplexities indicate that the topics are better. Fig. 2 shows the perplexities of topics 1 to 14. The optimal number of topics was, therefore, identified as 14.

Next, the topic terms were grouped into 14 topics. This discovery takes time and is resource intensive. The 14 topics and their top 10 terms were written in a text file, after which the script paused for expert inference of the topics based on the topic terms.

Sentiment Analysis: To answer RQ4, we utilized DistilBERT, a pre-trained machine learning model, to discover developer sentiments. DistilBERT was chosen as it is a smaller and more efficient version of BERT that classifies text as negative or positive.

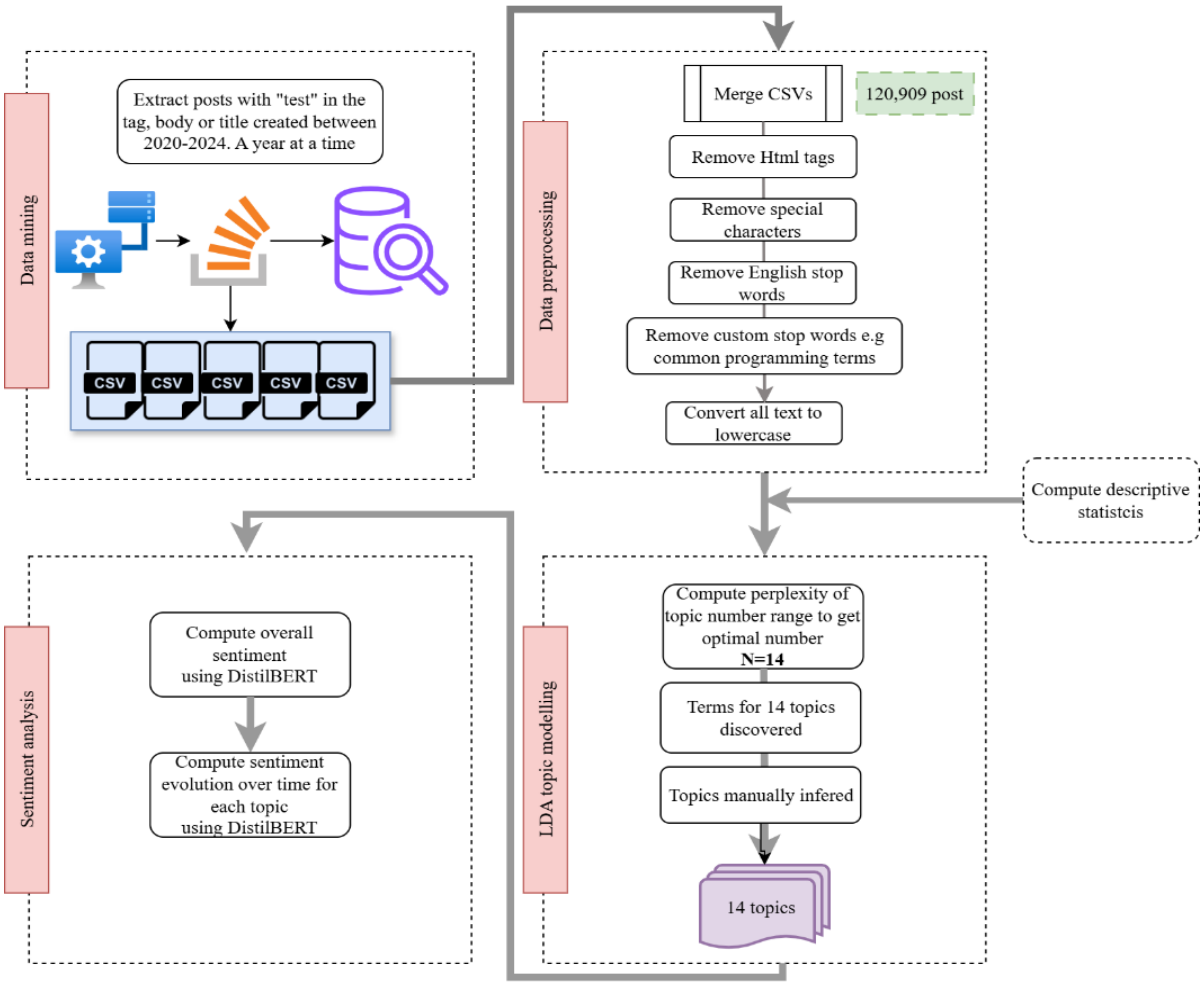

Fig. 1 Study methodology

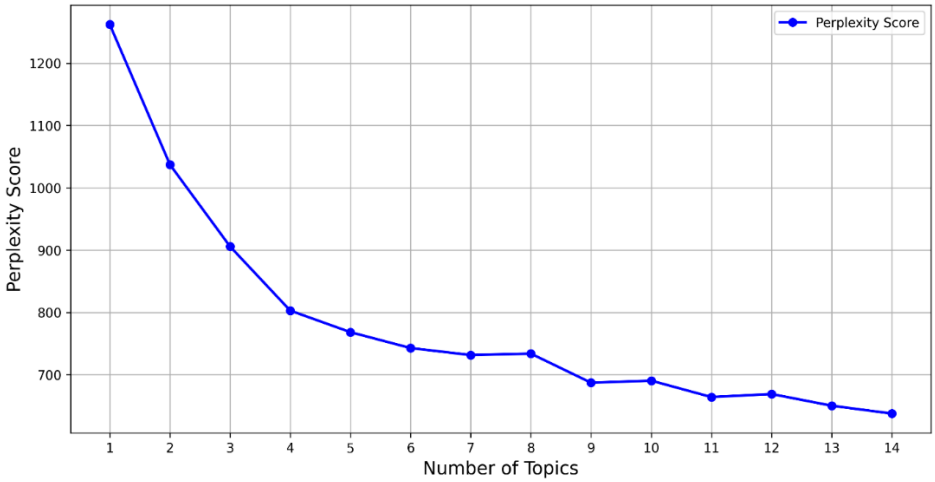

Fig. 2 Computation of topic perplexity

IV. RESULTS

This section presents results of the study in the order of the research questions defined in the Introduction section.

A. Topic and themes in software testing discussion (RQ1)

The analysis of 120,909 post revealed that 14 key topics related to software testing developers engaged in over the 5-year period under study. Table 1 shows the terms under each topic and the inferred topics.

Table 1 Topics terms and inferred topic

Topic	Topic Terms	Inferred Topic
Topic 1	junit, springframework, spring, internal, gradle, execute, com, jupiter, junit jupiter, hierarchical	JUnit Testing
Topic 2	model, unit, train, ve, pass, param, works, fail, dataset, params	Machine Learning Testing
Topic 3	android, com, image, event, google, https, ui, playwright, device, window	Android UI Testing
Topic 4	self, py, python, pytest, db, packages, django, models, session, init	Python Testing
Topic 5	js, react, jest, expect, modules, await, async, cypress, export, render	JavaScript Testing
Topic 6	query, select, form, group, sql, row, vue, max, rows, record	SQL Testing
Topic 7	mock, val, df, unit, mockito, fun, jmeter, patch, csv, implementation	Mock Testing
Topic 8	key, flutter, dart, child, color, hello, widget, await, src, async	Flutter Testing
Topic 9	angular, let, include, std, fixture, func, core, char, cmake, cpp	Angular Testing
Topic 10	request, api, client, json, password, message, token, username, await, async	API Testing
Topic 11	size, index, report, long, random, np, total, range, price, loop	Performance Testing
Topic 12	driver, page, task, selenium, browser, chrome, users, log, microsoft, options	Selenium Testing
Topic 13	maven, info, apache, testng, xml, io, net, configuration, scope, com	Maven Testing
Topic 14	build, config, src, docker, github, env, install, local, environment, home	Docker Testing

Fig. 3 is a word cloud that illustrates words that dominate in software testing discussions among developers on Stack Overflow. Terms like unit, mock, request, fail and message appear prominently indicating that unit testing, mocking HTTP request and failure handling are major concerts in testing discussions. Popular tools and technologies used in testing include JUnit, Jest and Cypress.

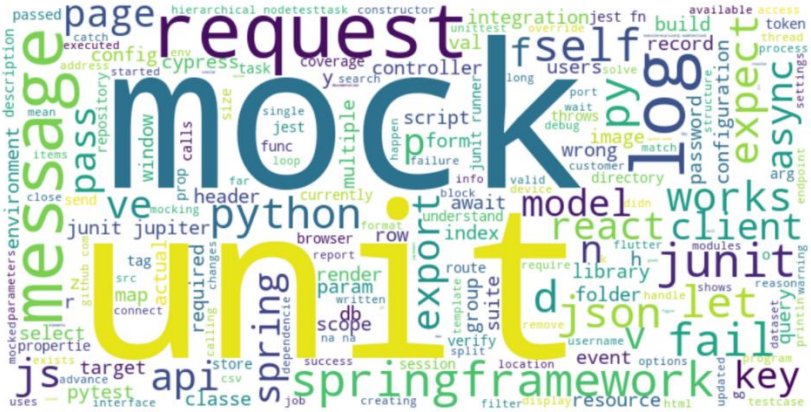

Fig. 3. Keywords and topics word cloud

Fig.4 illustrates how the topic terms are mapped to the specific identified topics. The network image depicts the terms or keywords that were used to identify the themes/ topic that form software testing discussion on SO.

As shown in Table 1, Fig. 3, and Fig. 4, discussions on Stack Overflow regarding developer software testing focus on the tools, methods, and various types of testing aimed at ensuring software functions reliably and securely. The identified topics cover areas such as: 1) Unit Testing, which checks individual components like functions or classes using tools such as JUnit and Pytest; 2) Machine Learning (ML) Testing, focused on validating the performance and accuracy of ML models through tasks like training, validation, and performance metrics; 3) Android UI Testing, ensuring that mobile UI, particularly for Android, functions properly; 4) Python Testing, which uses frameworks like Pytest for unit testing and applications based on Django; 5) JavaScript Testing, covering frameworks such as React with tools like Jest; 6) Web Application Testing, which ensures that web apps work across different browsers, handle loads, and address performance and security issues; 7) Security Testing, aimed at identifying vulnerabilities and employing measures like authentication and penetration testing; 8) Database Testing, to ensure that data retrieval, integrity, and performance are up to expectations; 9) Continuous Integration (CI) Testing, ensuring that code changes in CI pipelines are tested before being integrated into the project; and 10) Test Automation, which focuses on automating repetitive tasks for faster, more efficient testing, using tools like Selenium for web testing.

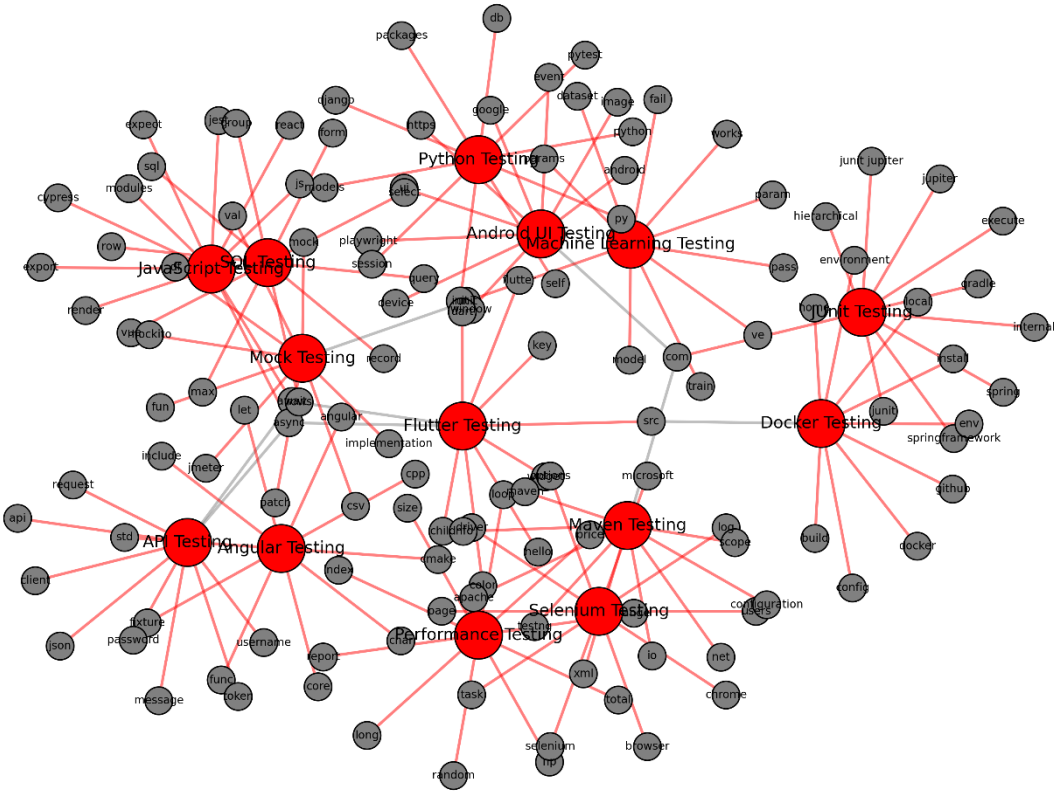

Fig. 4. Network graph of keywords and topics

A. Evolution of topics and trends in software testing discussions (RQ2)

As seen in Fig. 5, there is a general decline in the number of questions across all topics over time. Discussions related to machine learning, while remaining among the top topics and the highest in 2020, have steadily decreased through to 2024. JavaScript discussions were also prominent, surpassing machine learning discussions in mid-2021, but like ML, they also show a decline over the years. Android and Flutter discussions have both experienced a gradual decrease, with Android maintaining more posts than Flutter, which has the least number of posts. JUnit has shown relative stability, though a gradual decline in posts is still noticeable. Topics like Maven and Docker testing, though having fewer posts compared to machine learning, also follow a downward trend over time. Notably, for most topics, sharper declines can be observed starting from 2022.

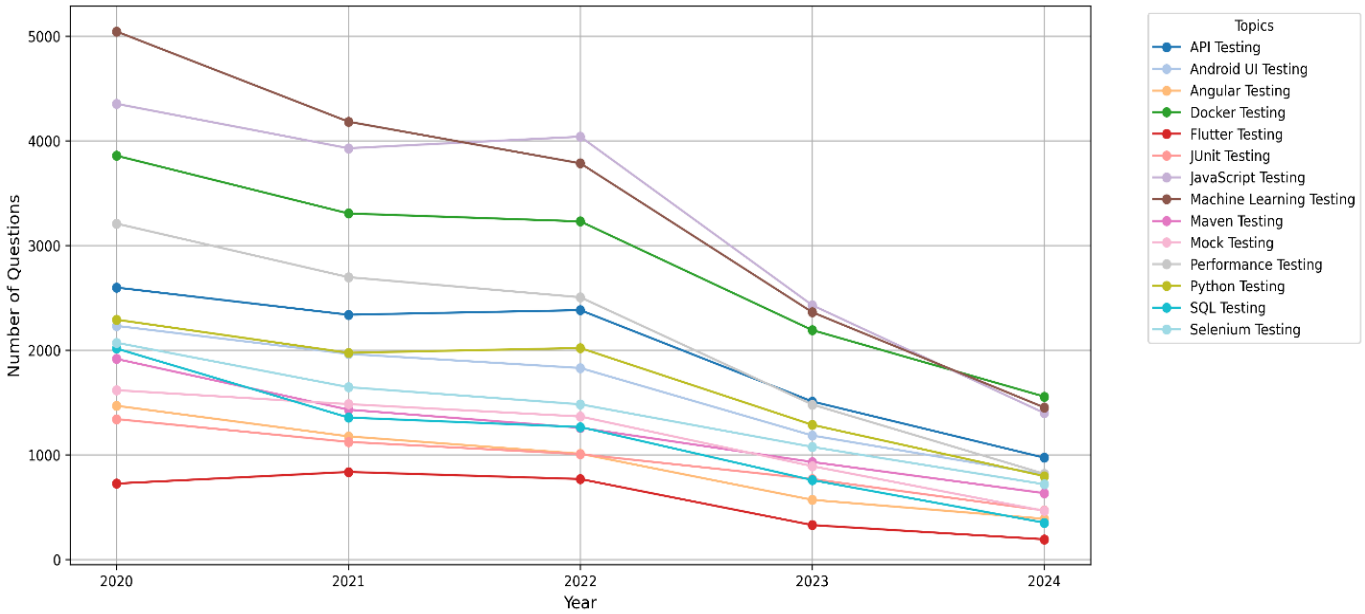

Fig. 5 Software testing topics and trends evolution. Data source: Authors' analysis of Stack Overflow data.

The trends mentioned above are clearly depicted in the heatmap in Fig. 5, with darker shades of blue indicating the intensity of discussions around each topic. In 2020, machine learning testing dominated, followed by JavaScript testing and Docker testing, in that order. This trend continues through 2024, with Docker testing eventually surpassing both machine learning testing and JavaScript testing.

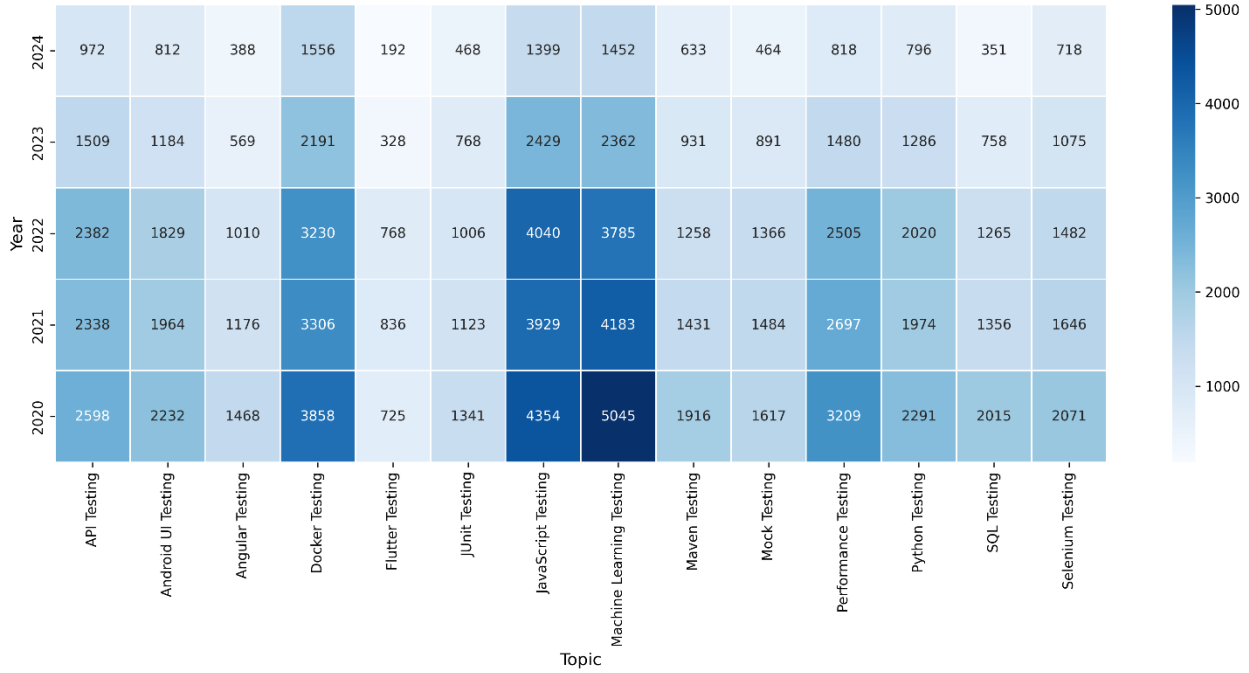

Fig. 6 Software testing topics and trends evolution heatmap. Data source: Authors' analysis of Stack Overflow data.

B. Developer engagement with software testing discussions (RQ3)

This study highlights a decline in developer engagement in software testing-related discussions on Stack Overflow. As shown in Table 2, in 2020, there were a total of 34,740 posts, with 27,922 (80.4%) receiving answers and 14,279 (41.1%) having accepted answers. In sharp contrast, by 2024, the total number of posts dropped to 11,019, with only 6,074 (55.1%) answered and just 2,571 (23.3%) accepted answers. These findings reflect a significant reduction in engagement.

Table 2. Software testing topics and trends statistics

Fig. 7 illustrates the declining trend in developer engagement with software testing topics. By 2023, the number of unanswered questions starts to rise, surpassing the number of accepted answers and approaching the number of answered questions. This suggests that, soon, for every two questions posted, only one is likely to receive an answer.

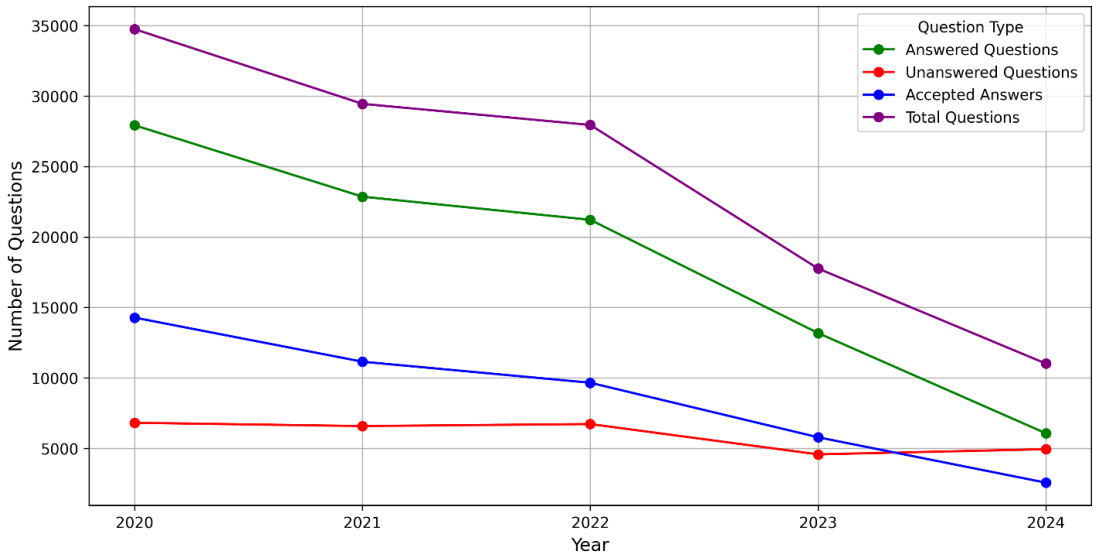

Fig. 7. Software testing questions status

C. Developer sentiment analysis (RQ4)

Regarding developer sentiment in software testing discussions, we observe that the majority of developers express negative sentiments. In 2020, 99.07% of all posts had negative sentiments, with only 0.93% reflecting positive sentiments, details of the breakdown are shown in Table 3. A similar pattern is evident across the years under study, continuing through to 2024, with a noticeable decline in the percentage of positive sentiments over the years. Specifically, positive sentiment dropped from 0.93% in 2020 to 0.55% in 2024.

The heatmap in Fig. 8 illustrates the moods of developers as they engage in discussions about software testing across different topics. Overall, developers predominantly express negative sentiments. In 2020, the most negative sentiments were observed from developers involved in Flutter Testing, followed closely by JUnit Testing and Angular Testing. By 2024, Flutter Testing continued to generate the most negative sentiments, followed by SQL Testing. Developers engaged in JavaScript and Machine Learning Testing showed notably positive sentiments, with Machine Learning Testing leading. However, the positive sentiments decline over the years, especially by 2024. API Testing displayed almost neutral sentiments, which gradually turned slightly negative by 2024. Overall, there has been a decline in positive sentiments across all topics. By 2024, the least negative sentiments were seen

from developers engaged in Machine Learning Testing and JavaScript Testing.

Table 3. Developer sentiment analysis

Year	Negative (N)	Positive (N)	Negative (%)	Positive (%)
2020	34418	322	99.07	0.93
2021	29218	225	99.24	0.76
2022	27741	205	99.27	0.73
2023	17653	108	99.39	0.61
2024	10958	61	99.45	0.55

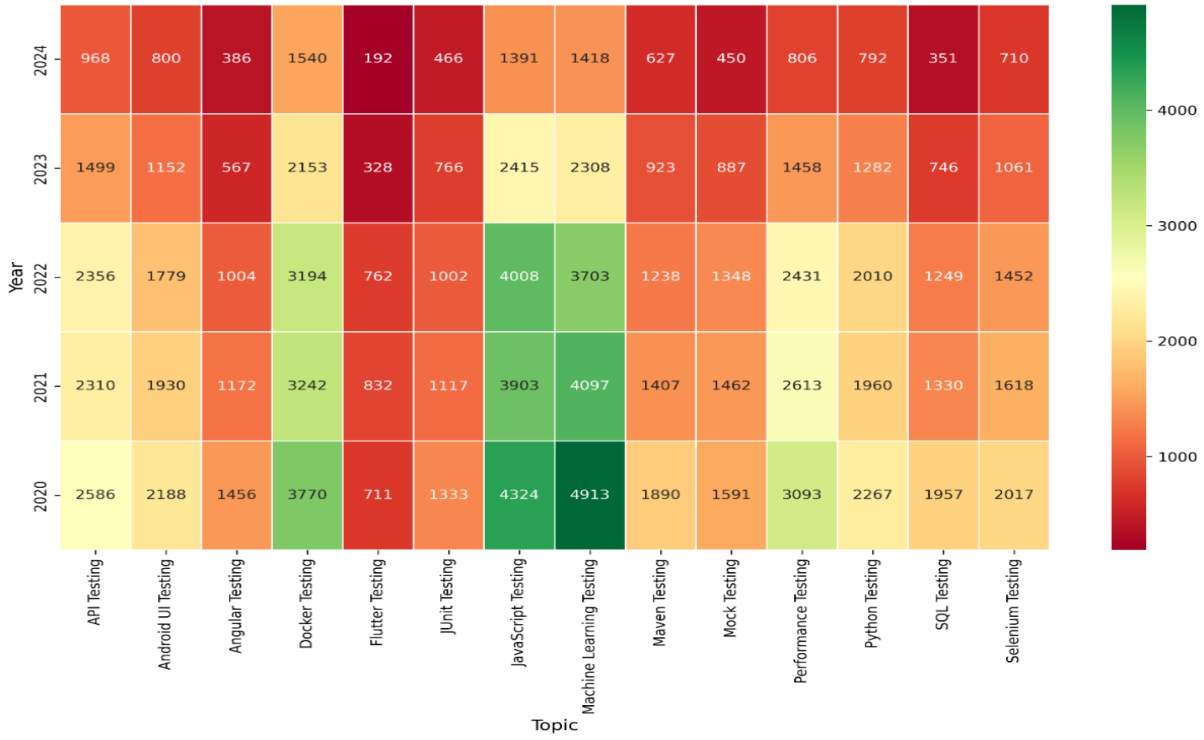

Fig. 8 Topic sentiment evolution heatmap

V. Discussion

In this section we present a discussion about the results presented in the Results section and their implications based on the research questions.

A. Topics and themes in software testing discussions

The analysis of 120,909 posts posted from 2020 to 2025 revealed 14 most dominant software testing themes and topics including Unit testing, Machine learning testing, Android UI testing, Python testing, JavaScript testing, SQL testing, mock testing, Flutter testing, Angular testing, API testing, performance testing, Selenium testing, maven testing and docker testing. The prominence is tools and frameworks such as Pytest, JUnit, Selenium and Web driver is also noticeable. These areas indicate issues that developers struggle with during testing. The presence of machine learning is critical as it shows that developers are adopting the use of machine learning for testing as well as testing their models before deploying them. Selenium testing is evidence that developers are automating tests, especially web testing. It is critical to note that developers experience challenges testing ML models and automating tests. Further, the presence of Android and Flutter indicates the continued adoption of mobile applications by industry and that developers are struggling to test mobile applications especially the user interfaces given that mobile applications need to adapt to varied screen factors to offer a seamless user experience across devices. The dominance of JavaScript with discussion centering around React and Jest indicates that developers grapple with testing web applications. Discussion

on security testing indicates developers are researching and facing challenges on how to test applications for vulnerabilities at a time where security and integrity of data and applications is important.

A. Evolution of topics and trends in software testing

For all the topics identified, a sharp decline in developer interest is noticeable, this more especially from 2022. The release of ChatGPT in 2022, could have contributed to the sudden decrease in the number of posts across all the topics [22]. A study by Kabir, et al. [8] found that students and novice programmers are often met with rude comments on their posts, accused of asking basic questions that reflect no effort. Naturally, such people would subsequently opt for ChatGPT as their source of answers since the system would not judge or be rude to them. Machine learning posts remain dominant as from 2024, indicating the industry's shift to use of MLs for testing but also a reflection that developers struggle to test their models. Flutter has the fewest testing posts owing to the fact it is a slightly new language, the discussions remain steady, showing continued interest up to the year 2022. The prominence of Docker speaks to the industry's shift to DevOps where development and operations are integrated, this seems to be an areas developer are struggling with.

B. Developer engagement in software testing

This study has revealed a significant decline in developer engagement in software testing discussions. The number of questions posted has decreased, while the number of unanswered and unaccepted answer questions has risen. We attribute this shift to the rise of ChatGPT, where developers can receive personalized support and even have code snippets generated. While ChatGPT offers valuable assistance, Q&A forums have long served as a rich resource for researchers seeking to understand developer behavior. However, based on the current trend, this resource may be lost, as ChatGPT does not maintain conversations for others to refer to. Instead, it generates new responses for each query without retaining past interactions.

C. Developer sentiments in software testing discussions

Our findings indicate that most developers express negative sentiments in their software testing discussions. A study by Swillus and Zaidman (2023) also found that negative sentiments dominate on Stack Overflow, with 63 out of 108 posts (58%) classified as negative. While it is understandable that developers may visit Q&A platforms seeking help and might already be stressed, our analysis of sentiment by topic and its evolution over time reveals that some topics provoke more negative sentiments than others. Notably, Flutter generates the most negative sentiments. Although the negative sentiment associated with Flutter has decreased from 2020 to 2024, it remains the leading topic in terms of developer frustration. This could be attributed to the challenges developers face when learning to test a relatively new language, as Flutter was only introduced in 2017. Our previous study (Wambua, 2024) highlighted the areas where Flutter developers struggle.

Interestingly, developers working on Machine Learning (ML) models in areas like training and datasets, tend to exhibit positive sentiment, although this shifts to a more negative tone towards 2024. This change could be due to the initial excitement surrounding ML testing, which gradually gives way to frustration as developers encounter challenges. On the other hand, topics like API Testing elicit almost neutral sentiments, suggesting that these issues may not be as burdensome for developers. This could be because API testing technologies have been around for longer, and as a result, more developers are familiar with them.

VI. CONCLUSION

This study analyzed software testing discussions on Stack Overflow for a five-year period, uncovering key topics, trends, developer engagement, and sentiments. Our findings reveal that discussions on Stack Overflow predominantly revolve around testing tools, frameworks, and methods, with JUnit, machine learning, Docker, and JavaScript emerging as dominant themes. However, we observed a notable decline in the volume of posts and developer engagement over time, particularly after 2022, which corresponds with the increasing use of AI tools like ChatGPT for problem-solving. This decline, coupled with a rising number of unanswered questions, indicates a shift in how developers seek assistance for software testing challenges.

The sentiment analysis also highlighted a generally negative sentiment surrounding software testing topics. This is particularly evident in mobile testing (e.g., Flutter) where developers expressed frustration over testing challenges. Despite some positive sentiment around machine learning discussions in earlier years, the overall tone of software testing conversations on Stack Overflow has grown increasingly negative over time.

These findings have several important implications. The decline in developer engagement suggests that while traditional Q&A forums like Stack Overflow have been invaluable for peer support, their role may be diminishing as AI-powered tools provide instant, personalized responses. This shift calls for further exploration into the interplay between AI tools and developer community platforms, and how the former might be integrated into or replace aspects of the latter. Disappearance of Q&A forums will deny the research community a mine where they can study developer behaviour.

For future research, we recommend investigating the underlying causes of the decline in developer engagement, particularly in relation to the influence of AI tools like ChatGPT. Additionally, there is potential to further explore how sentiment analysis can enhance the software testing process by providing insights into developer frustrations, allowing for the development of tools that can address these challenges more effectively. For example, Stack Overflow can have sentiment labels that communicate developer

feelings based on their posts. Lastly, given the increasing adoption of machine learning and DevOps, it is critical to explore these areas further to understand the evolving needs of developers and the challenges they face in testing these new technologies.

In conclusion, while the study sheds light on key trends and challenges in software testing, it also underscores the importance of ongoing research to adapt to the shifting landscape of software development, testing, and developer support systems. By understanding developers' engagement and sentiments, the software engineering community can develop more effective solutions, tools, and platforms to enhance the software testing process.

REFERENCES

[1] A. Salahirad, G. Gay, and E. Mohammadi, "Mapping the structure and evolution of software testing research over the past three decades," *Journal of Systems and Software*, vol. 195, p. 111518, 2023/01/01/ 2023, doi: 10.1016/j.jss.2022.111518.

[2] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software Testing With Large Language Models: Survey, Landscape, and Vision," *IEEE Transactions on Software Engineering*, vol. 50, no. 4, pp. 911-936, 2024, doi: 10.1109/TSE.2024.3368208.

[3] D. Chevers, *A Software Development Approach for Driving Competitiveness in Small Firms*. Auerbach Publications, 2023.

[4] I. Santos, E. F. Coutinho, and S. R. S. Souza, "Software testing ecosystems insights and research opportunities," presented at the Proceedings of the XXXIV Brazilian Symposium on Software Engineering, Natal, Brazil, 2020. [Online]. Available: https://doi.org/10.1145/3422392.3422458.

[5] F. S. Ahmed, A. Majeed, T. A. Khan, and S. N. Bhatti, "Value-based cost-cognizant test case prioritization for regression testing," *PLOS ONE*, vol. 17, no. 5, p. e0264972, 2022, doi: 10.1371/journal.pone.0264972.

[6] R. Blanco, M. Trinidad, M. J. Suárez-Cabal, A. Calderón, M. Ruiz, and J. Tuya, "Can gamification help in software testing education? Findings from an empirical study," *Journal of Systems and Software*, vol. 200, p. 111647, 2023/06/01/ 2023, doi: 10.1016/j.jss.2023.111647.

[7] Y. Wu, J. Su, D. D. Moran, and C. D. Near, "Automated software testing starting from static analysis: current state of the art," *arXiv preprint arXiv:2301.06215*, 2023, doi: 10.48550/arXiv.2301.06215.

[8] S. Kabir, D. N. Udo-Imeh, B. Kou, and T. Zhang, "Is Stack Overflow Obsolete? An Empirical Study of the Characteristics of ChatGPT Answers to Stack Overflow Questions," presented at the Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, 2024. [Online]. Available: https://doi.org/10.1145/3613904.3642596.

[9] A. M. Alghamdi, W. Aljedaani, H. Jalali, S. Ludi, and M. M. Eler, "Understanding developer challenges and trends in web accessibility: a stack overflow analysis," *Universal Access in the Information Society*, 2024/11/19 2024, doi: 10.1007/s10209-024-01174-3.

[10] T. Varun Kumar, "A Comprehensive Empirical Study Determining Practitioners' Views on Docker Development Difficulties: Stack Overflow Analysis," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 12, no. 1, pp. 157-164, 2024.

[11] J. He *et al.*, "Representation Learning for Stack Overflow Posts: How Far Are We?," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 3, p. Article 69, 2024, doi: 10.1145/3635711.

[12] S. Izzat and N. N. Saleem, "Software testing techniques and tools: A review," *Journal of Education and Science*, vol. 32, no. 2, pp. 30.0-44.0, 2023, doi: 10.33899/edusj.2023.137480.1305.

[13] H. M. Ali, M. Y. Hamza, and T. A. Rashid, "A comprehensive study on automated testing with the software lifecycle," *arXiv preprint arXiv:2405.01608*, 2024, doi: 10.48550/arXiv.2405.01608.

[14] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation," *IEEE Transactions on Software Engineering*, vol. 50, no. 1, pp. 85-105, 2024, doi: 10.1109/TSE.2023.3334955.

[15] M. A. Zmuda, "A Software Framework Based on JUnit for Automated Software Testing in Computer Science Courses," in *2024 IEEE Frontiers in Education Conference (FIE)*, 13-16 Oct. 2024 2024, pp. 1-8, doi: 10.1109/FIE61694.2024.10892903.

[16] V. Yadav, R. K. Botchway, R. Senkerik, and Z. K. Oplatkova, "Robotic automation of software testing from a machine learning viewpoint," in *Mendel*, 2021, vol. 27, no. 2, pp. 68-73, doi: 10.13164/mendel.2021.k.0e3.

[17] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software testing with large language models: Survey, landscape, and vision," *IEEE Transactions on Software Engineering*, 2024, doi: 10.1109/TSE.2024.3368208.

[18] D. Elsner, D. Bertagnolli, A. Pretschner, and R. Klaus, "Challenges in regression test selection for end-to-end testing of microservice-based software systems," presented at the Proceedings of the 3rd ACM/IEEE International Conference on Automation of Software Test, Pittsburgh, Pennsylvania, 2022. [Online]. Available: https://doi.org/10.1145/3524481.3527217.

[19] R. Santos, I. Santos, C. Magalhaes, and R. d. S. Santos, "Are We Testing or Being Tested? Exploring the Practical Applications of Large Language Models in Software Testing," in *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, 27-31 May 2024 2024, pp. 353-360, doi: 10.1109/ICST60714.2024.00039.

[20] M. Boukhlif, N. Kharmoum, and M. Hanine, "LLMs for Intelligent Software Testing: A Comparative Study," presented at the Proceedings of the 7th International Conference on Networking, Intelligent Systems and Security, Meknes, AA, Morocco, 2024. [Online]. Available: https://doi.org/10.1145/3659677.3659749.

[21] N. Weeraddana, M. Alfadel, and S. McIntosh, "Characterizing Timeout Builds in Continuous Integration," *IEEE Transactions on Software Engineering*, vol. 50, no. 6, pp. 1450-1463, 2024, doi: 10.1109/TSE.2024.3387840.

[22] A. W. Wambua, "What Do Flutter Developers Ask About? An Empirical Study on Stack Overflow Posts," *Journal of Software Engineering Research and Development*, vol. 12, no. 1, pp. 7: 1-7: 13, 2024, doi: 10.5753/jserd.2024.3620.

[23] A. Alanazi and R. Alfayez, "What is discussed about Flutter on Stack Overflow (SO) question-and-answer (Q&A)

website: An empirical study," *Journal of Systems and Software*, vol. 215, p. 112089, 2024, doi: 10.1016/j.jss.2024.112089.

[24] C. C. Silva, M. Galster, and F. Gilson, "Applying short text topic models to instant messaging communication of software developers," *Journal of Systems and Software*, vol. 216, p. 112111, 2024/10/01/ 2024, doi: https://doi.org/10.1016/j.jss.2024.112111.

[25] A. Abdelrazek, Y. Eid, E. Gawish, W. Medhat, and A. Hassan, "Topic modeling algorithms and applications: A survey," *Information Systems*, vol. 112, p. 102131, 2023/02/01/ 2023, doi: https://doi.org/10.1016/j.is.2022.102131.

[26] N. Prakash, H. Wang, N. K. Hoang, M. S. Hee, and R. K.-W. Lee, "PromptMTopic: Unsupervised Multimodal Topic Modeling of Memes using Large Language Models," presented at the Proceedings of the 31st ACM International Conference on Multimedia, Ottawa ON, Canada, 2023. [Online]. Available: https://doi.org/10.1145/3581783.3613836.

[27] J. Zimmermann, L. E. Champagne, J. M. Dickens, and B. T. Hazen, "Approaches to improve preprocessing for Latent Dirichlet Allocation topic modeling," *Decision Support Systems*, vol. 185, p. 114310, 2024/10/01/ 2024, doi: https://doi.org/10.1016/j.dss.2024.114310.

[28] B.-X. Du and G.-Y. Liu, "Topic analysis in lda based on keywords selection," *Journal of Computers*, vol. 32, no. 4, pp. 1-12, 2021, doi: 10.53106/199115992021083204001.

[29] C. Treude and M. Wagner, "Predicting good configurations for GitHub and stack overflow topic models," presented at the Proceedings of the 16th International Conference on Mining Software Repositories, Montreal, Quebec, Canada, 2019. [Online]. Available: https://doi.org/10.1109/MSR.2019.00022.

[30] A. Pereira, K. Gama, C. Zimmerle, and F. Castor, "Reactive Programming with Swift Combine: An Analysis of Problems Faced by Developers on Stack Overflow," in *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*, 2023, pp. 109-115.

[31] M. U. Haque, L. H. Iwaya, and M. A. Babar, "Challenges in docker development: A large-scale study using stack overflow," in *Proceedings of the 14th ACM/IEEE international symposium on empirical software engineering and measurement (ESEM)*, 2020, pp. 1-11, doi: 10.1145/3382494.3410693.

[32] M. Wankhade, A. C. S. Rao, and C. Kulkarni, "A survey on sentiment analysis methods, applications, and challenges," *Artificial Intelligence Review*, vol. 55, no. 7, pp. 5731-5780, 2022, doi: 10.1007/s10462-022-10144-1.

[33] K. L. Tan, C. P. Lee, and K. M. Lim, "A Survey of Sentiment Analysis: Approaches, Datasets, and Future Research," *Applied Sciences*, vol. 13, no. 7, p. 4550, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/7/4550.