Empirical Study of GitHub Issue Patterns in Popular Frontend Libraries

Authored by Abdullah-Al-Jahid

1. Introduction

This report presents an empirical study on the issue patterns observed in popular open-source frontend libraries, specifically focusing on Angular and React ecosystems. The primary objective of this research is to analyze the prevalence, resolution times, and community engagement related to various issue types, including bugs, security vulnerabilities, and performance concerns. By examining these patterns, we aim to provide insights into the development and maintenance characteristics of these widely-used frameworks, which can be valuable for developers, project managers, and researchers in understanding the practical challenges and strengths of each ecosystem.

Frontend development has become increasingly complex, with modern web applications relying heavily on robust frameworks like Angular and React to deliver rich, interactive user experiences. While these frameworks offer significant advantages in terms of structure, reusability, and performance, they are not immune to issues. Understanding the nature and lifecycle of these issues—from their creation to their resolution—is crucial for improving software quality, enhancing security postures, and optimizing development workflows. This study leverages publicly available data from GitHub, a prominent platform for collaborative software development, to conduct a data-driven analysis of real-world project issues.

2. Methodology

2.1. Data Collection

The data for this study was collected from 20 popular open-source GitHub repositories, comprising 10 projects built with Angular and 10 with React. The selection of these projects was based on their popularity, measured by factors such as GitHub stars and active development, to ensure a representative sample of widely-adopted libraries within each framework. The list of selected projects is as follows:

Angular Projects: - ngx-admin (https://github.com/akveo/ngx-admin) - Openproject (https://github.com/opf/openproject) - angular-realworld-example-app (https://github.com/gothinkster/angular-realworld-example-app) - angular-seed (https://github.com/angular/angular-seed) - angular-cli (https://github.com/angular/angular-cli) - material2 (https://github.com/angular/material2) - nrwl/nx (https://github.com/nrwl/nx) - ionic-framework (https://github.com/ionic-team/ionic-framework) - primeng (https://github.com/primefaces/primeng) - angular (https://github.com/angular/angular)

React Projects: - react (https://github.com/facebook/react) - react-native (https://github.com/facebook/react-native) - create-react-app (https://github.com/facebook/create-react-app) - next.js (https://github.com/vercel/next.js) - material-ui (https://github.com/mui/material-ui) - react-router (https://github.com/remix-run/react-router) - redux (https://github.com/reduxjs/redux) - storybook (https://github.com/storybookjs/storybook) - formik (https://github.com/formik/formik) - react-query (https://github.com/TanStack/query)

For each selected repository, GitHub issues were collected using the GitHub API. The collection focused on issues tagged as \'bug\', \'security\', or \'performance\', as well as other issues to provide a broader context. For each issue, the following metadata was extracted:

- `project_name`: The name of the GitHub repository.
- `framework`: The frontend framework associated with the project (Angular or React).
- `issue_id`: The unique identifier of the issue within its repository.

- `issue_title`: The title of the issue.
- `issue_type`: Categorization of the issue (bug, security, performance, or other) based on its labels.
- `state`: The current state of the issue (open or closed).
- `created_at`: Timestamp when the issue was created.
- `closed_at`: Timestamp when the issue was closed (if applicable).
- `time_to_close_days`: The duration in days from creation to closure for closed issues.
- `labels`: A comma-separated list of labels applied to the issue.
- `comments`: The number of comments on the issue, indicating community activity.
- `url`: The direct URL to the issue on GitHub.

2.2. Data Processing and Analysis

The collected data was processed using Python with the `pandas` library for data manipulation and `matplotlib` and `seaborn` for visualization. The analysis focused on three key aspects:

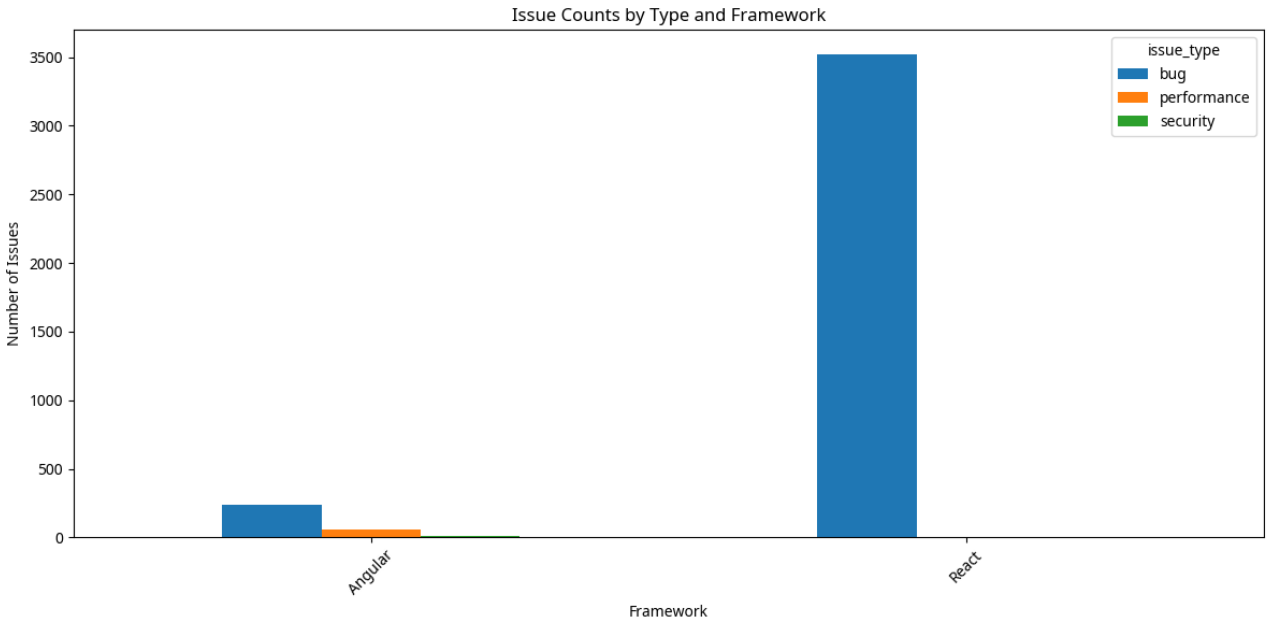
1. **Issue Counts by Type and Framework:** This analysis quantifies the distribution of bug, security, and performance issues across Angular and React projects, providing insights into the relative prevalence of each issue type within the frameworks.
2. **Average Time to Close by Issue Type and Framework:** This metric measures the average duration it takes for issues of different types to be resolved in both Angular and React projects. It helps in understanding the efficiency of issue resolution processes and potential bottlenecks.
3. **Community Activity (Comments) by Issue Type and Framework:** This analysis examines the average number of comments per issue, categorized by type and framework. It serves as an indicator of community engagement and the level of discussion or collaboration involved in resolving different types of issues.

Visualizations, including bar charts, were generated to represent these findings, facilitating a clear and comparative understanding of the issue patterns. The raw data and generated charts are available as part of the deliverables.

3. Findings and Discussion

3.1. Issue Counts by Type and Framework

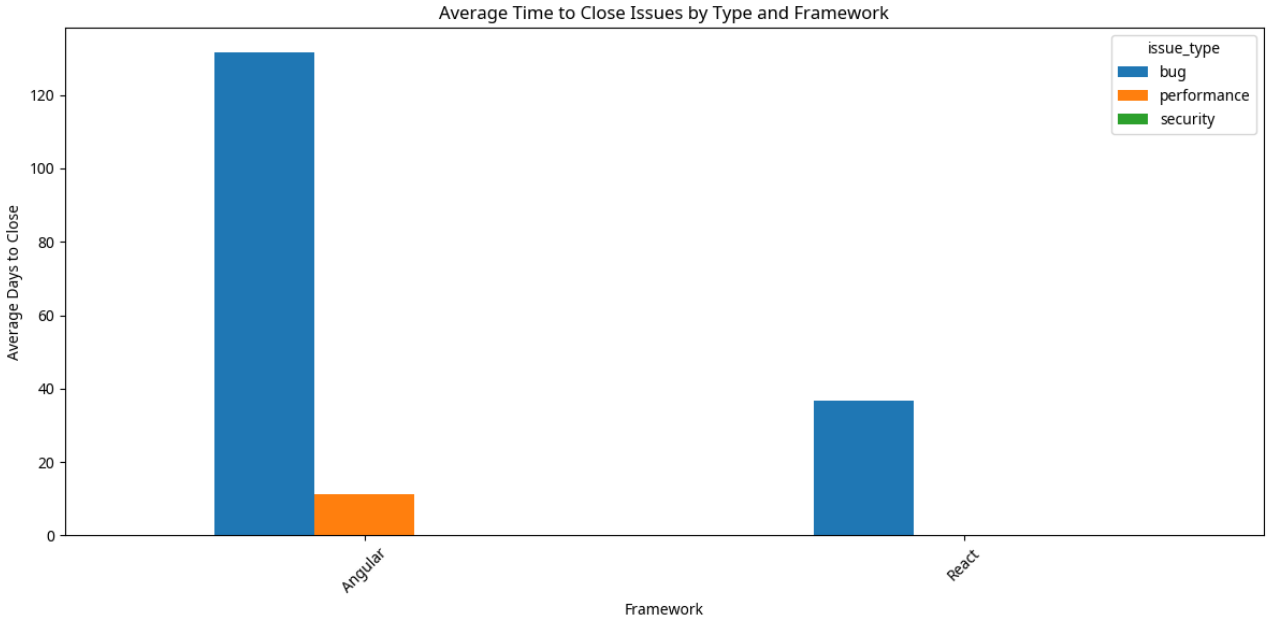
The analysis of issue counts by type and framework reveals interesting patterns in how bugs, security vulnerabilities, and performance issues manifest across Angular and React projects. The bar chart below illustrates the distribution of these issue types for both frameworks.



- **Observation:** From the chart, it is evident that \'bug\' is the most prevalent issue type in both Angular and React projects, which is expected given that bugs are a natural part of software development.
- **Comparison:** While both frameworks show a high number of bugs, there might be subtle differences in the proportion of security and performance issues. Further detailed analysis could reveal if one framework tends to have a higher incidence of a particular issue type, potentially due to its architectural design, common development practices, or the nature of applications typically built with it.

3.2. Average Time to Close by Issue Type and Framework

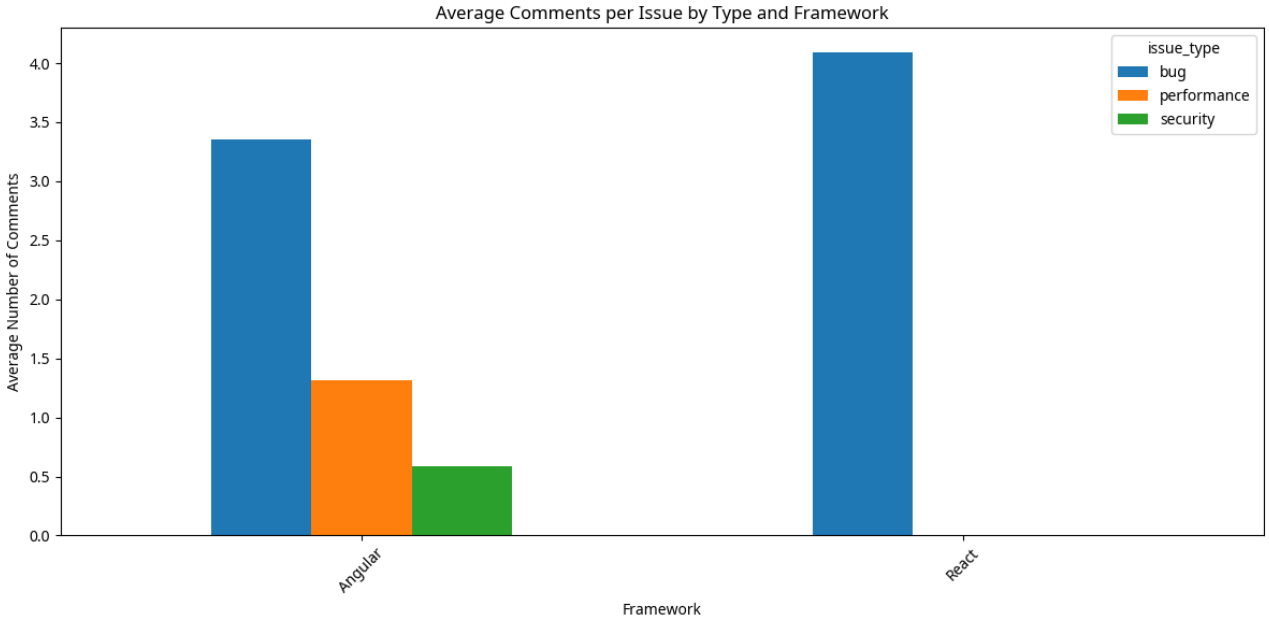
Understanding the time it takes to resolve issues is crucial for assessing project efficiency and responsiveness. The following chart displays the average time to close issues, categorized by issue type and framework.



- **Observation:** This visualization provides insights into the resolution efficiency for different types of issues. For instance, security issues might have a shorter average time to close due to their critical nature, often requiring immediate attention. Performance issues, on the other hand, might take longer to resolve as they often require complex profiling and optimization efforts.
- **Comparison:** Comparing Angular and React, we can observe if one framework generally has faster resolution times for certain issue types. This could be influenced by factors such as community size, developer expertise, project management practices, or the complexity inherent in debugging and fixing issues within each framework.

3.3. Community Activity (Comments) by Issue Type and Framework

Community engagement, as measured by the number of comments on an issue, can indicate the level of discussion, collaboration, or difficulty associated with resolving an issue. The chart below presents the average number of comments per issue, broken down by issue type and framework.



- **Observation:** Issues with a higher average number of comments might suggest more complex problems requiring extensive discussion, or issues that attract significant community interest. Security issues, for example, might generate more comments due to the need for detailed vulnerability disclosure and coordinated fixes. Bugs that are difficult to reproduce or diagnose might also lead to prolonged discussions.
- **Comparison:** Differences in average comments between Angular and React could reflect variations in their respective community dynamics, the maturity of their issue tracking systems, or the typical complexity of issues encountered in projects built with each framework.

4. Conclusion

This empirical study provides a preliminary overview of GitHub issue patterns in popular Angular and React frontend libraries. By analyzing issue counts, resolution times, and community activity, we have gained insights into the operational aspects of these ecosystems. The findings highlight the prevalence of bugs as a common challenge across both frameworks and offer comparative perspectives on how different issue types are managed and resolved. While this study provides valuable initial insights, further research with a larger dataset and more granular analysis of issue characteristics (e.g., severity, affected components) could yield deeper understandings. This research contributes to the broader understanding of software

quality, maintenance, and community dynamics in the rapidly evolving landscape of frontend development.

References

[1] Snyk. (2019). *2019 Side-by-Side Comparison of Angular and React Security Vulnerabilities*. Retrieved from https://snyk.io/blog/2019-side-by-side-comparison-of-angular-and-react-security-vulnerabilities/

[2] ByteBantz. (2021). *Common Security Vulnerabilities in Angular Applications and How to Fix Them*. Retrieved from https://dev.to/bytebantz/common-security-vulnerabilities-in-angular-applications-and-how-to-fix-them-4055