**Open-Source Research Report**

By Aarithi Rajendren

The open-source project I have decided to work on is Go language. This project has an active community with 125k stars, 17.8k forks, and 62,069 commits. This project became an open-source project in 2009 so, its age is 16 years old. The size of the project is anywhere from 10 to 20 MB depending on depending on the included dependencies and libraries. Go language is primarily a tool for developers since it is well-suited for projects involving distributed systems, web servers, data processing, and other backend components. Due to its simple, fast, and reliable characteristics many of the popular DevOps tools are written in Go such as Docker and Kubernetes. Additionally, Go is one of the best for cloud compatibility and allows developers to focus on the actual problem. Go is on track to becoming one of the largest, most active open-source projects and its community is supported by comprehensive documentation and fosters collaboration [1]. Also, it has improved system reliability due to its focus on static typing and built-in concurrency mechanisms to help developers write more reliable and robust applications [1].

A good practice for newcomers is to understand the project’s description and its contribution guidelines. This can be done by reading the README and the contribution guidelines website, which can all be found in the project repository on GitHub: <https://github.com/golang/go?tab=readme-ov-file>. Next, it is recommended that newcomers start working on issues with the good first issue tag since those issues are more straightforward and beginner friendly. Also, reviewing merged pull requests can give newcomers a good idea of what a successful contribution looks like. To learn the Go language, it is recommended to start with small projects and make use of Go’s standard library. A popular website among Go users to help learn Go is: <https://golangbot.com/learn-golang-series/> [1]. The key points in the Go community code of conduct are: treat everyone with respect and kindness, be thoughtful in how you communicate, don’t be destructive or inflammatory, if you encounter an issue send an email to [conduct@golang.org](mailto:conduct@golang.org) [2]. To contribute to the project via a pull request, these are the steps: first make a fork, then clone the fork, and then make a new branch to make modifications, commit those modifications, lastly submit it for a code review. In terms of code style, all Go source files must conform to format outputted by the gofmt tool [3]. Go source code uses camel case and there is no fixed line length [3]. In Go, names tend to be shorter than other programming languages and there should be consistency among lines of code that are near each other [3].

One of the starter issues in the Go open source project is called bufio.reader empty write. This issue states that when the reader interface in Go is wrapped with a buffer then there is an empty write being made even if there was data that needed to be read [4]. However, when the reader interface reads data from an input stream, such as stdin, the data is read a s expected. The significance of an empty write being made is that in client-server interactions using Unix sockets, the empty write results in an EOF (end of file) signal, so the server closes the data transmission connection before the data is completely sent and read [4]. The issue is trying to implement two different ways for input to be read from the user, one through an input stream and another through a buffer. For the issue to be fixed, the code needs to explicitly check if the buffer is empty, and if that is true, that would be the only case where an empty write should be made, otherwise the buffer should be read and written completely before ending with the EOF signal. Thus, I would estimate 4-7 days to completely resolve this issue depending on the developer’s coding familiarity with Go. Initially, the first 1-2 days would be to understand and become familiar with Go language and its packages. Following this, another 1-2 days would be to go through the code where the error might be occurring and add debug statements to narrow which functions/lines of code is causing the error. The subsequent 1-2 days would be to try test cases to make sure the changes to the code fixed the issue and if not, then there would have to be additional debugging and code changes. Finally, the last day would be for reviewing all the changes and following the Go documentation to send the fixed code to the project maintainer.

**References**

[1] Pavan Belagatti, “Why is Go so Damn Popular Among Developers,” *DEV Community*, Jun. 08, 2020. https://dev.to/pavanbelagatti/why-is-go-so-damn-popular-among-developers-2d6h (accessed Feb. 05, 2025).

[2] “Go Community Code of Conduct - The Go Programming Language,” *Go.dev*, 2025. https://go.dev/conduct (accessed Feb. 05, 2025).

[3] “Go style guide”, github.io, 2025. <https://google.github.io/styleguide/go/guide.html> (accessed Feb. 05, 2025).

[4] “bufio package - bufio - Go Packages,” *Go.dev*, 2025. https://pkg.go.dev/bufio (accessed Feb. 05, 2025).

‌