Hoppscotch: Rust Engineer Programming Challenge

Introduction

As part of your interview process as a Rust Engineer at Hoppscotch, you have to complete this take home assignment within 1 week of time (from when the challenge is sent to you) and make it public in a GitHub repository. The challenge is designed to test your understanding of Rust programming and your skills in adapting to and writing bridges of communication between JavaScript and Rust. Along with that, keen eye for performance and code quality is also considered and rewarded appropriately.

Problem Statement

You are required to build a ZIP file Viewer that can view the contents of a zip file in the system. You are to write this using the Tauri framework. The app needs to have the following functionality:

1. The app should provide a window where the user can click to select and open a ZIP file.
2. Once a ZIP file is selected, the app should parse the ZIP file and show a tree showing all the files in the ZIP file (just the path).
3. The app should handle ZIP files locked with a password and ask the user the password (along with its validation) and then show the files to the user.
4. The app should also show some metadata about the ZIP file itself (name, size, compressed size etc).
5. The app should have a welcome screen where it remembers and shows the list of last 5 ZIPs you have opened.

Constraints & Nuances

1. The ZIP file handling part should be written in Rust.
2. Any frontend JS framework (or even vanilla) can be used to build the frontend/ui portion of the app.
3. You should use Tauri Commands to communicate back and forth between the UI and the Rust layer.

Judging Criteria

1. We are not judging UI polish here, so you will not be receiving negative points for a bad UI, but a good UI will be given consideration as well.
2. Code cleanliness and formatting maintenance.
3. Following best practices for the corresponding platforms used.
4. Performance considerations and API design of the Rust-Frontend communication layer.
5. Following proper Git history and commit practices.
6. Documentation is considered an easy plus, documented code will be given bonus points.
7. Robustness and error handling will be important for crucial sections, but bonus points for further graceful error handling.

Good luck!