Title: GITS: An Advanced Approach to Streamlining Git Version Control

Team Members: Harikrishnan Venkatesh, Kruthik Jonnagaddala Thyagaraja, Prabhanjan Vinoda Bharadwaj, Sai Kishore Honnavalli Ravi Shankar.

Introduction

In the realm of version control systems, Git has emerged as the standard tool for developers, enabling efficient collaboration on projects. However, even proficient Git users acknowledge the intricacies and unintuitive nature of certain commands. This essay delves into the GITS project, an open-source tool aimed at refining Git's functionality and semantics. We will elucidate the core functionalities of GITS, focusing on its innovative features, namely GITS Profile, GITS Rebase, and GITS Undo. These features relieve some of the pain points associated with traditional git functionalities.

Things We Liked

Simplifying User Profiles with GITS Profile

An essential facet of the GITS project is the 'GITS Profile' command, engineered to facilitate seamless transitions between different user accounts. In multifaceted scenarios, users often manage both personal and enterprise profiles, requiring them to switch between profiles. The GITS Profile command addresses this need, enabling users with a straightforward mechanism to toggle between their designated accounts. This functionality not only accelerates workflow but also mitigates the probability of inadvertent commits under the incorrect profile.

Enhancing Rebasing with GITS Rebase

Rebasing constitutes a critical operation in Git, involving the relocation or consolidation of a series of commits onto a new base commit. The 'GITS Rebase' command aims to ease the process of rebase, often considered as a cumbersome and tedious process, to make it more intuitive and add more functionality in a single command, reducing the need to execute a series of commands.. Conventional Git rebasing requires a nuanced understanding of source and destination branches, often creating confusion among users. GITS Rebase, conversely, adopts a streamlined approach, enabling users to specify the destination branch through intuitive command line arguments. This refinement eschews ambiguity, guaranteeing a seamless rebase process, particularly valuable for advanced users orchestrating intricate branch management strategies.

Empowering Users with GITS Undo

A standout facet of GITS is the 'GITS Undo' command, an instrumental functionality that simplifies the process of reverting changes. In traditional Git workflows, developers commonly resort to 'git clean' or 'git revert', but these commands exhibit limitations. 'Git revert' undoes changes across all files, potentially culminating in unintended consequences. Conversely, 'git checkout --filename' proves to be nonintuitive. GITS Undo empowers users to swiftly and efficiently revert specific changes, affording a level of precision and control that was hitherto unavailable in Git. This command proves invaluable when recent changes do not align with project objectives, allowing for a seamless return to the last stable commit.

The Vision and Implementation of GITS

The idea of GITS stemmed from the need to simplify common version control commands, ultimately enhancing the overall version control experience. In addition to introducing novel commands for streamlined version control, the GITS repository encapsulates all the standard functions of Git, encompassing push, pull, commit, and more. The installation process was straightforward using the requirements.txt file included in the repository.

Challenges we Faced

While implementing GITS, a challenge was understanding the need for this particular tool. Initially, there was an absence of evident drawbacks in traditional Git that necessitated the introduction of GITS. It was only upon delving into the individual command documents that the true potential and functionality of GITS became clear. It is worth noting that the creators of the repository could enhance user experience by including a comprehensive documentation comparing the commands in Git and GITS, providing users with a clear reference for transitioning between the two systems, making it easier for users to adopt GITS as their standard version control tool.

Learnings

Understanding the Codebase

The GITS repository had a modularised approach towards its structure. Several commands were internally calling other commands. We had to understand the structure of the code first before we dove into the individual commands. This gave us a macroscopic view of the code and the reason for the way the code was compartmentalized.

Evaluating a Repository.

We understood the importance of test cases and code coverage for a repository. Along with evaluating the functionalities, it's necessary to evaluate how the repository covers edge cases, and the understandability of the code. We also got essential practice in evaluating a repository on various components such as commit contributions, ability to raise issues, and many other metrics.