Harrison Che
CS411
10/4/24

Agile

Completing user stories:

1. As a vanilla git power-user that has never seen GiggleGit before, I want the features to be simple to access so that I will be able to quickly understand how to use GiggleGit as well as easily use the system.
2. As a team lead onboarding an experienced GiggleGit user, I want to efficiently onboard them and add them to our repos so that they can get to work immediately under the GiggleGit version control system.

Create a third user story:

As a GiggleGit user who works on a Windows machine and on a Mac, I want seamless compatibility between the two machines so I can work at home and in the office.

- Task: Create multi-platform compatibility
- Ticket #1: Configuration and installation of GiggleGit
  A comprehensive and concise process should be created to set up GiggleGit on MacOS and Windows. Installation and configuration should be standardized.
- Ticket #2: Synchronization of GiggleGit across machines
  A system should be put in place so that users can synchronize their GiggleGit configs (like setting up remotes) through an account or SSH keys. This should work on a small-scale/personal setting as well as for large projects where entire teams will work on a project.

Issues with the last user story:

The phrase, “As a user I want to be able to authenticate on a new machine,” is not a user story as it doesn’t describe the outcome/ experience that a user would want. Moreover, it’s simply too vague and should be considered more as a feature/ requirement of GiggleGit. A proper user story should include a goal, or more importantly, what they’re trying to achieve.

Formal Requirements

List one goal and one non-goal:

Goal: SnickerSync is should provide seamless synchronization for the user's version control conflicts with a playful touch as opposed to "merging in silence"

Non-goal: SnickerSync should not develop a system that is entirely unrelated to GiggleGit and is not compatible with the version control system.

Non-Functional Requirements:

Security of User Data: SnickerSync must protect user data and ensure that no unauthorized access to version histories or private repositories occurs.

Functional Requirement #1: Encryption of Merge Data

Implement encryption protocols for all data being synced via SnickerSync, ensuring that version histories and any relevant metadata are securely transmitted and stored.

Functional Requirement #2: Authentication for Accessing SnickerSync

Require users to authenticate with their GiggleGit credentials before initiating a sync operation, ensuring only authorized users can access the their features.

Random Assignment for User Studies: The system should support random assignment of users into control groups and variants to ensure unbiased results in user studies of SnickerSync.

Functional Requirement #3: Automated User Group Assignment

During testing, implement a feature that automatically assigns users into control groups and experimental groups when they engage with SnickerSync, ensuring randomization.

Functional Requirement #4: User Study Tracking and Logging

Develop a logging system that tracks user interactions, indicating which group they were assigned to, what actions they performed, and how they responded to the SnickerSync interface. The gathered data will be used for evaluating study results.