Project description:

CodeChuckle is introducing a new diff tool: SnickerSync—why merge in silence when you can sync with a snicker? The PMs have a solid understanding of what it means to "sync with a snicker" and now they want to run some user studies. Your team has already created a vanilla interface capable of syncing with the base GiggleGit packages.

Project requirements

Goal: Implement a method of merging through SnickerSync that allows users to merge their code with an element of humor while upholding the GiggleGit integrity.
Non-Goal: Changing the core merging process of GiggleGit apart from the superficial SnickerSync.
Non-functional requirement 1: Access Security

Functional requirement 1: Implement restrictions to SnickerSync's management capabilities that PM's can access.

Functional requirement 2: Track user actions and changes on SnickerSync in a log.

Non-functional requirement 2: Responsive

Functional requirement 1: Ensure users get feedback durin the syncing process with minimal response times.

Functional requirement 2: Ensure the system can uphold a large concurrent number of users without loosing ground on response times.

Agile

Theme: Get GiggleGit demo into a stable enough alpha to start onboarding some adventurous clients
Epic: Onboarding experience
User story 1: As a vanilla git power-user that has never seen GiggleGit before, I want to use similar commands to git so that the learning process is expedited

Task: Seamless Git user adoption

- Ticket 1: Command wrapping

  Commonly used Git commands should be wrapped and aliased in GiggleGit for an easy transition to GiggleGit.

- Ticket 2: Help command

There should be a GiggleGit help command where git users can find the parallel GiggleGit to git commands. This feature should show common commands in Git with a detailed explanation of how to produce the same functioning command.

User story 2: As a team lead onboarding an experienced GiggleGit user, I want to quickly be able to monitor their workflow

Task: Traceable and clear workflow

- Ticket 1: Workflow logs

  Create a feature that allows team leads to view an activity log of a user's recent actions and commands in GiggleGit.

- Ticket 2: Workflow summary

  Provide an overview of the user's profile and preface their preferred workflow and common strategies.

User story 3: As a developer joining a new team, I want to be able to access files and understand the documentation structure with ease.

Task: Repository organization and folder structure visibility

- Ticket 1: Search and display features

  Create a feature that allows users to look at a display of the folders and repository structure. Create a feature that allows users to search up the addresses of certain files or folders.

- Ticket 2: Important file descriptions

  Allow annotations for important file folders and directories so that users know how to better navigate the repositories.

This is not a user story. Why not? What is it?

- As a user I want to be able to authenticate on a new machine

This is not a user story because it is very vague. User stories should provide what kind of user the user writing the story is. The story provides a needed action but in order for it to be a useful user story they should provide why the action is needed so that the relevant task can be efficiently implemented. The sentence is more similar to a broad requirement and possible task.

Dependency Diagrams