Agile

CodeChuckle is a startup whose product is GiggleGit, a version control system “where merges are managed by memes." (It saddens me to say that this was a joke written by ChatGPT for 131)

You have just been hired as employee number n for some small number n. They have the dev chops to make a demo, but you are their first serious developer.

Here is a theme and an epic:

- 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 understand what GiggleGit is and how it can serve as a better alternative to vanilla Git, so that i can evaluate its benefits and decide if it’s worth using for my projects.
    - Task: Create a guide that explains what GiggleGit is and how it compares to vanilla Git.
        - Ticket 1: Create an introductory guide to GiggleGit
            - We need to Write a clear and detailed guide comparing Git and GiggleGit features. Mention GiggleGit meme based merging system.
        - Ticket 2: Develop an interactive tutorial to display Giggle Git’s unique features and advantages over git.
            - We need to Design an interactive tutorial that introduces Giggle Git’s core features and concepts like meme-based merges and how it adds value compared to Git.

- User Story 2: As a team lead onboarding an experienced Giggle Git user, i want a quick onboarding process so new employees can adapt and work right away.
    - Task: Create a streamlined onboarding process for experienced Giggle Git users.
        - Ticket 1: Create Documentation for Giggle Git
            - Create detailed Documentation covering all aspects of Giggle Git. It should include instructions on installation, configuration,

basic commands troubleshooting and more. The documentation should cater to both new users and experienced Git users, highlighting the unique features of GiggleGit while ensuring it is accessible and informative for all levels of users.

        - Ticket 2: Create a onboarding guide for new Employees experienced in Git
            - Develop a guide that goes through step by step on getting started with Giggle Git. The guide should help new Employees create accounts, connecting repositories, how to use features and ect.

- User Story 3: As a developer, I want an easy way to resolve merge conflicts in Giggle Git.
    - Task: Create a streamlined merge conflict resolution process while still leveraging GiggleGit's meme merge.
        - Ticket 1: Design a merge conflict resolution interface
            - Create a user friendly interface in GiggleGit to help resolve merge conflicts by showing conflicting changes, and simple control system to help users keep changes they wish to keep and integrate memes.
        - Ticket 2: Implement an automatic suggestion to resolve merge conflicts.
            - Implement AI that analyzes the conflicting changes and recommends possible solutions allowing the developer to quickly apply a fix manually fix it themselves with the guidance of the suggestions.

Formal Requirements

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.

- Goal: Create an entertaining and humerios merging system by allowing users to add memes which in this case Sync with a snicker as they merge to bring a humorous and entertaining merging experience without sacrificing functionality.
- Non-Goal: Add meme suggestions based on the day of the week.
- Non-functional requirement: User Access and Security
    - Functional requirement:
        - Users without permission can’t change or modify snickering concepts only use and access.Need to keep permission strict
        - Only Users within the same project are allowed to use the SnickerSync concept and no one outside the project should access it. Implement Protected Routes.
- Non-functional requirement User Study with Random Assignment:
    - Functional requirement:
        - Implement a system that random selects users and puts them into a control group or experimental group to ensure accurate results
        - Implement an automotive system that keeps track of the experimental group and control group and track their activity and interaction with Snicker Sync and return the data.