SnickerSync Requirements

Project Description:
SnickerSync introduces a new way to handle diffs and merges in version control—“why merge in silence when you can sync with a snicker?” The system aims to create a more engaging and interactive experience for developers by adding humorous elements to the merge process. It is built on top of the base GiggleGit packages, with a user interface that can integrate syncing in an entertaining way.

Project Requirements
Goal: Create an engaging and entertaining merging experience that enhances user satisfaction by incorporating humorous elements during merge conflicts and diff reviews, such as the good o’ snicker

Non-Goal: Provide in-depth code analysis or advanced debugging features as part of the merge process.

Non-Functional Requirement 1: Security and Access Control

- **Functional Requirements:**
    1. Ensure that only authorized users, such as developers within the same project, can initiate or approve SnickerSync merge operations.
    2. Implement role-based access control to allow PMs to review and modify different snickering concepts before they are made available to end users.

Non-Functional Requirement 2: Concept Flexibility and Adaptation

- **Functional Requirements:**
    1. Develop an intuitive interface for PMs that allows them to create, edit, and categorize different snickering concepts (ex: puns, visual gags, meme styles) in a way that ensures a very diverse library of humor types can be expanded over time.
    2. Implement an adaptive humor engine that automatically suggests new snickering concepts to PMs based on user engagement data (ex: the number of smiles recorded via emoji reactions) to stay on trend and keep the merging experience fresh and amusing for users!

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 be able to understand the unique features of GiggleGit and how they compare to traditional Git, so that I can decide whether it meets my needs.

- **Task: Create GiggleGit introductory guide for Git power-users**
    - **Ticket 1:** Develop user guide for Git power-users
        - **Details:** Create a detailed guide that highlights the main features of GiggleGit, compares its functionality to vanilla Git, and emphasizes the unique aspects like the amazing meme-based merges.
    - **Ticket 2: Create a set of comparison examples**
        - **Details:** Implement example scenarios that illustrate the typical Git workflows vs how they would be handled in GiggleGit, focusing on key differences.

User Story 2: As a team lead onboarding an experienced GiggleGit user, I want to provide easy to understand documentation and a streamlined process for new team members, so that they can be productive with GiggleGit as quickly as possible.

Task: Develop team onboarding documentation.

Ticket 1: Create onboarding checklist for new users

- **Details:** Design a checklist that outlines steps for new team members, including account setup, basic commands, and how to perform meme-based merges
-

Ticket 2: Implement an onboarding wizard in the UI

- **Details:** Develop a simple onboarding wizard that guides new users through essential first steps in GiggleGit, such as connecting to repositories and practicing merges.

User Story 3: As a user, I want to receive a fun and engaging introduction to GiggleGit, so that I feel extra motivated and excited to explore its features.

Task: Develop an interactive and GiggleGit tutorial!

Ticket 1: Design a meme-based onboarding tutorial!

- **Details:** Create an engaging tutorial that uses memes to explain core features of GiggleGit, making the learning process entertaining and memorable for users.

Ticket 2: Implement fun games and exercises for key actions and features!

- **Details:** Within the tutorial, users can create a sample repository and resolve merges, with step-by-step instructions and memes coming left and right.

The statement “As a user I want to be able to authenticate on a new machine” is not a complete user story because it lacks context and value, especially the “so that” part, which is supposed to explain the reason or benefit behind the user’s goal. Like why does the user want to authenticate in a new machine? What is the purpose for that? Without those answers, it is not a user story.