Prompt: 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: Enable users to understand and implement SnickerSync, while maintaining an engaging experience that demonstrates “syncing with a snicker”.
* Non-Goal: Create advanced customization features for syncing because the primary focus is studying user interactions with SnickerSync’s main features
* Non-functional requirement 1: Accessibility
  + Functional requirements:
    - Implement a user authentication system so that only registered users can use SnickerSync
    - Develop an admin interface allowing PM access control where they assign roles, manage user permissions, etc.
* Non-functional requirement 2: Randomization
  + Functional requirements:
    - Implement data logging to identify and analyze trends
    - Use a random group assignment algorithm to ensure that the study sample is unbiased

Prompt: 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.

**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 differences between vanilla git and GiggleGit, and be able to switch between the commands without difficulty.

Task: Develop support for easy transitioning

Ticket 1: Create a guide with documentation

Ticket 2: Implement a help function that provides information on command switching

User Story 2: As a team lead onboarding an experienced GiggleGit user, I want to be able to track and manage my team’s GiggleGit usage.

Task: Create team usage tracking features

Ticket 1: Include a feature that allows for role assignments

Ticket 2: Build an activity dashboard

User Story 3: As a developer who is part of a team, I want to be able to use GiggleGit when collaborating on projects.

Task: Build collaboration tools

Ticket 1: Allow real-time coediting so that multiple users can work simultaneously

Ticket 2: Include commenting features

“As a user, I want to be able to authenticate on a new machine” is not a user story as it does not include a benefit. As stated in class, “benefits bring money, not features.” This statement is more of a requirement statement indicating that the user requires that the authentication process works on a new machine, allowing them to securely access GiggleGit from multiple devices, ensuring continuity and security.