1. 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

Complete the following tasks:

Complete these user stories:

As a vanilla git power-user that has never seen GiggleGit before, I want to understand how GiggleGit differs from Git so I can decide if I want to incorporate it into my team’s workflow.

As a team lead onboarding an experienced GiggleGit user, I want to ensure that my team can quickly integrate into our team’s specific use of GiggleGit so they can be productive without friction

Create a third user story, one task for this user story, and two associated tickets.

As a developer evaluating GiggleGit for enterprise use, I want to understand its scalability and compatibility with our existing infrastructure so I can determine if it meets our organization’s needs.

Task: Evaluate GiggleGit’s scalability and compatibility.

Tickets:

- Write a comparison document showcasing how GiggleGit handles version control differently from Git
- Test GiggleGit in a large repository and assess its performance

Tasks should be a single phrase. (As should themes and epics. See those provided.)

User stories should be one to three sentences.

Tickets should have a title consisting of a single phrase and details that are long enough to sufficiently describe what needs to be done. You do not need to assign points to the tickets

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 does not explain why the user needs this feature or how it benefits the workflow. Rather than a user story, this corresponds closer to a requirement or feature request.

2. 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.

Complete the following tasks:

List one goal and one non-goal

- **Goal:** *Provide a seamless experience by incorporating the 'snickering' concept to improve user feedback and collaboration.*

- **Non-Goal:** *This release will not implement a new version control system beyond the existing GiggleGit packages.*

Create two non-functional requirements. Here are suggestions of things to think about:

Who has access to what

PMs need to be able to maintain the different snickering concepts

A user study needs to have random assignments of users between control groups and variants

1. *Ensure only authorized users can modify snickering concepts and access sensitive data.*

2. *Ensure that user studies are conducted in a controlled and repeatable manner, maintaining data integrity.*

For each non-functional requirement, create two functional requirements (for a grand total of four functional requirements).

1.1. *Implement role-based access control (RBAC) so that only authorized PMs can modify snickering concepts.*
1.2. *Use OAuth to securely log-in users and enforce permissions.*

2.1*. Implement an algorithm for random assignment of users into control and experimental groups.*
2.2 *Store experimental data in a database to prevent duplicate entries and ensure consistency in test groups.*