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:

1. Complete these user stories:
    - As a vanilla git power-user that has never seen GiggleGit before, I want to quickly learn how to use it
    - As a team lead onboarding an experienced GiggleGit user, I want to help them understand our team’s workflow
2. Create a third user story, one task for this user story, and two associated tickets.
    - 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

    - User story 3: As a frequent user of GiggleGit, I want to be delighted by a variety of memes so that I never get bored
        - Task: Have GiggleGit store several thousand memes
            - Ticket 1: Crawl the internet for memes
                a. Use Python soup to crawl Reddit, Instagram, and Twitter for memes
            - Ticket 2: Set up a database to store all those memes
                a. Find and purchase the database
                b. Use SQL and a backend framework to retrieve and store data
3. 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
        - The benefit to the user is not clear enough, so it should be a task instead

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:

1. List one goal and one non-goal
2. 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
3. For each non-functional requirement, create two functional requirements (for a grand total of four functional requirements).

Goal: Create a web application that sends a joke every time the user syncs a repository

Non-goal: Detect when a user syncs, and draw from a random pile of jokes in a database to send to them

Non-functional requirement 1: Variability

- Functional requirements:
  - Have a database large enough to store thousands of jokes
  - Write an algorithm to ensure that the user does not get the same jokes

Non-functional requirement 2: Security

- Functional requirements:
  - Require that users log into their Github account, but encrypt that information
  - Allow users to remove their data from CodeChuckle if they don’t wish to use the service anymore