Complete the following tasks:

1. Complete these user stories:
   - As a vanilla git power-user that has never seen GiggleGit before, I want to customize the meme's I'm able to see, to make it a more enjoyable experience to work on projects.
   - As a team lead onboarding an experienced GiggleGit user, I want to access my colleague's work progress, so I can understand the appealing aspects of such a version control system.

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

     USER STORY: As a new programmer, I want to know each aspect of the VCS, so that I may understand how to navigate it easily.

     TASK: Create an easy to follow, comprehensive tour of the VCS that allows new users to feel comfortable using it.

     TICKETS:
     1. Outline: List critical aspects to showcase to new users, along with details and examples of how to use the feature.
     2. DOD LATJER
   - 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
2. 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
     No explanation as to why the user requests this feature, and no clarification on what kind of user.

# 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:

1. List one goal and one non-goal

   GOAL: Add capability to compare two branches that the user chooses, storing any changes found.

   NONGOAL:  Create an easy to use interface so the user can navigate through and organize the changes.

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).

   Non-functional Requirements:

   Security:

   Functional requirements:

   - Provide training and resources on security practices to keep data safe.

- Enforce access controls where only users can only access the recources they have permission to.

Performance:

-Must have support to minimize runtime for all operating environments

-SnickerSync performance should remain relatively similar even when there are large amounts of users at a given time.