Agile

1) Complete these user stories:
As a vanilla git power-user that has never seen GiggleGit before, I want to be able to use GiggleGit with little to no steep learning curve in order to create merges with memes so I can have fun viewing the memes while coding.

As a team lead onboarding an experienced GiggleGit user, I want to be capable of having the team learn the product fast and easily so we can all have fun viewing the memes to boost team morale.

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: As a new GiggleGit user who has constantly changing deadlines, I want to be able to use a mood or urgency indicator in order to document the urgency/importance of a merge.

Task: Enable GiggleGit to request an emotion or urgency level to influence the meme

Ticket 1: Create an easy-to-use UI to allow the user to input a feeling

- We need to make sure that the user cannot influence the meme's outcome (limit the response to emotions/feelings)

Ticket 2: Implement the ability for GiggleGit to know which memes have been used already to avoid duplicates

- Create a database or list of meme names that have been used to ensure that merges are not producing the same boring outcomes

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.

This is not a user story because this needs a clear benefit. Although it follows the format of a specific actor and some action, there is seemingly no benefit, making it not a user story. It is instead possibly just a suggestion written by the user for their wishes.

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**
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).**

1) Goal: This tool aims to boost morale and improve productivity by incorporating humor into the boring syncing process.
Non-goal: By running user studies, PMs want to collect data to understand how others react to this new tool and whether or not it can make users happier with a little sense of humor.

2) Non-functional requirements:
Non-functional requirement 1: Accessibility

3) Functional requirements:
- Only PMs should have access to the ability to change what is heard or seen by the user
- Users should be able to merge and commit without any disruptions but not access to change the tools active

Non-functional requirement 2: Scalability

3) Functional requirements:
- A certain number of users should be placed into control groups and another group should be given access to the tool to determine the effectiveness
- Continue to add or remove people from the user studies to ensure scalability. This will determine if this tool is useful/dependable across a growing number of users.