Agile:

1. Complete these user stories:
    - As a vanilla git power-user that has never seen GiggleGit before, I want to intuitively understand how to use the version control system in order to streamline the transition to GiggleGit.
    - As a team lead onboarding an experienced GiggleGit user, I want to see all my previous merges through a timeline of the memes I used at the time in order to reference them easily.
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 complete git novice who knows nothing about git and git commits, I want to learn the intricacies of git through popular internet culture like memes to understand merges and git as a whole.

Task: Introduce git through memes

Ticket 1: Add a "Getting Started Section"

Add a section that is presented to users that succinctly explains the nature of git through memes to facilitate understanding for first time users.

Ticket 2: Implement an optional chatbot

Implement an LLM chatbot that will explain through memes why the user may be having trouble with their merge/commits.

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 for two main reasons, it doesn't provide a clear type of user it just claims to be a general one. And there is never an explanation for why they want to authenticate on a new machine (what benefit would it bring?). This is more of a functional specification, just a demand that needs to be fulfilled.

Formal Requirements:

- Goal: Create a diff tool capable of working with GiggleGit to ease the merging and file comparison process more so than just vanilla GiggleGit.
- Non-goal: Use integrated AI to automatically create merges and syncs with the push of a button.
- Non-functional requirement 1: Reliability
    - Functional Requirements:
        - Automatic updating to merges and other GiggleGit uses, the user must be able to see the changes as soon as they occur, within 5 seconds at most.

        - SnickerSync must be available to use with GiggleGit whenever on any machine, if it is down it should be for an hour at most.

- Non-functional requirement 2: Usability
    - Functional Requirements:
        - The file changes should be shown in a side-by-side view with consistent color coding.
        - The user should be able to easily access past commits and merges, they should all be intuitive to find.