Agile:

Theme: Get GiggleGit demo into a stable enough alpha to start onboarding some adventurous clients
Epic: Onboarding experience

User Story 1:

Title: Explore GiggleGit for the first time
Acceptance Test: testNewUserCanExploreGiggleGit
Priority: High
Story Points: 5
Description: As a vanilla git power-user that has never seen GiggleGit before, I want to easily navigate the user interface, access essential commands, and familiarize myself with key features without extensive documentation.

User Story 2:

Title: Onboard experienced GiggleGit user quickly
Acceptance Test: testTeamLeadCanOnboardExperiencedUser
Priority: Medium
Story Points: 3
Description: As a team lead onboarding an experienced GiggleGit user, I want to ensure that the setup process is streamlined so they can contribute to our workflow with minimal downtime.

User Story 3:

Title: Authenticate securely in GiggleGit
Acceptance Test: testUserCanAuthenticateSecurely
Priority: High
Story Points: 8
Description: As a user, I want to securely authenticate within GiggleGit to ensure that my repositories and workflows remain secure and I can switch machines easily.

Task for User Story 3:

Title: Implement secure authentication method

Tickets for User Story 3:

Ticket 1:
Title: Design secure authentication flow
Details: Define a secure authentication workflow that includes multi-factor authentication and token-based sessions, ensuring compatibility with third-party identity providers.

Ticket 2:
Title: Implement token-based authentication for new users
Details: Develop the backend to generate and manage authentication tokens for new users, ensuring that these tokens are encrypted and comply with security best practices.

Non-User Story Analysis:

As a user I want to be able to authenticate on a new machine

This is **not a user story** because it lacks context or a clear reason behind the need. A user story should explain the "why" behind the action. To make it a proper user story, it needs to include the motivation or benefit for the user. For example, "As a user, I want to authenticate on a new machine so that I can access my repositories without needing to reconfigure everything."

Formal Requirements

Goal and Non-Goal

Goal:
To evaluate the usability and effectiveness of SnickerSync during the user study, ensuring that participants can seamlessly perform git operations while syncing with a snicker.

Non-Goal:
Optimizing the underlying Git merge performance is **not** the focus of this study. The emphasis is on evaluating user interaction with SnickerSync's interface and user experience.

Non-Functional Requirements

1. **Access Control**:
   Only PMs and authorized stakeholders should have access to maintain the different snickering concepts and manage configurations related to the SnickerSync tool.
2. **Random Assignment for User Study**:
   Users participating in the study must be randomly assigned to control groups (using the vanilla Git diff tool) and variant groups (using SnickerSync) for unbiased results.

Functional Requirements

For **Non-Functional Requirement 1** (Access Control):

- **Functional Requirement 1.1**: The system should allow PMs to log in using their unique credentials and gain access to the SnickerSync management interface.
- **Functional Requirement 1.2**: The system should enforce role-based permissions, ensuring that only users with PM or admin roles can view or modify snickering concepts.

For **Non-Functional Requirement 2** (Random Assignment for User Study):

- **Functional Requirement 2.1**: The user study platform should automatically assign each new participant to either the control or variant group using a randomization algorithm.
- **Functional Requirement 2.2**: The system should track and log each user's group assignment and ensure that the randomization is fair and unbiased across the study’s duration.