GiggleGit Requirements (User Stories, Tasks, Tickets)

Theme:

Get GiggleGit demo into a stable enough alpha to start onboarding some adventurous clients

Epic:

Onboarding experience

User Stories and Tasks

User Story 1:

As a vanilla Git power-user that has never seen GiggleGit before, I want to quickly understand its differences from Git so that I can evaluate whether it's useful for my workflow.

Task: Improve documentation and onboarding guide

Ticket 1: Write clear documentation for Git users

- Compare GiggleGit's workflow to Git with a side-by-side example
- Write a quick start guide focusing on common Git workflows

Ticket 2: Implement interactive onboarding prompts

- Guide new users through setting up GiggleGit with step-by-step CLI prompts
- Provide hints when users encounter unfamiliar commands

User Story 2:

As a team lead onboarding an experienced GiggleGit user, I want to ensure my team follows best practices so that collaboration is smooth.

Task: Establish collaboration guidelines

Ticket 1: Create best practice documentation for teams

- Define branch management, merging strategies, and resolving conflicts
- Provide case studies on effective collaboration

Ticket 2: Implement role-based permissions in GiggleGit

- Allow team leads to configure access levels for different contributors
- Ensure only authorized users can approve merges

User Story 3 (New Requirement)

As a developer experimenting with GiggleGit, I want a sandbox environment so that I can test its features without affecting my real repositories.

Task: Implement a sandbox mode

Ticket 1: Create an isolated sandbox environment

- Allow users to try GiggleGit features in a simulated repo
- Ensure changes made in the sandbox don't affect actual repositories

Ticket 2: Provide pre-loaded test data

- Populate the sandbox with sample commits, branches, and merge conflicts
- Help users understand key features by experimenting with real-world scenarios

SnickerSync Requirements (Goals, Non-Goals, Functional & Non-Functional Requirements)

Goal:

Enable developers to resolve code differences collaboratively using meme-based conflict resolution.

Non-Goal:

Automatically generate memes based on code changes.

Non-Functional Requirements & Functional Requirements

Non-Functional Requirement 1: Security

Ensure that only authorized users can access and modify SnickerSync repositories.

Functional Requirements:

- **Authentication & Authorization:** Users must authenticate via OAuth before accessing SnickerSync.
- **Role-Based Access Control (RBAC):** Only authorized users can initiate a merge, and permissions are enforced via team settings.

Non-Functional Requirement 2: Experimentation & User Studies

Allow PMs to conduct A/B testing on different meme-based merge strategies.

Functional Requirements:

- **Randomized User Assignment:** Users are randomly assigned to different merge conflict resolution strategies.
- **Variant Logging & Feedback Collection:** Merge experiences are logged, and users can provide feedback on meme effectiveness.