User Stories

1. As a vanilla git power-user that has never seen GiggleGit before, I want to be able to perform all the basic git operations (clone, commit, merge, etc.) without confusion, so I can feel comfortable using it right away.

Task:

- Ensure basic git functionality is intuitive for first-time users.

Tickets:

- **Ticket 1 Title**: "Simplify Clone, Commit, and Push UI"
    - **Details**: Redesign the user interface for the most common git operations (clone, commit, push) to match the expectations of seasoned git users. Make sure buttons, commands, and actions are labeled clearly and mirror git commands.
- **Ticket 2 Title**: "Create a Git Basics Quickstart Guide"
    - **Details**: Write concise documentation that walks users through setting up GiggleGit, performing a clone, making a commit, and pushing to a repository. The guide should cater to experienced git users with minimal introduction but highlight any notable differences in GiggleGit.

---

2. As a team lead onboarding an experienced GiggleGit user, I want to be able to quickly invite and manage new users to my team, so they can start working with the same repositories and have access to relevant features.

Task:

- Implement team management features for inviting new users.

Tickets:

- **Ticket 1 Title**: "Create Team Invitation System"
    - **Details**: Build a feature that allows team leads to invite new users via email. Ensure that the invitee receives an email with a secure link to join the team and that the lead can track the status of pending invites.
- **Ticket 2 Title**: "Add Role-Based Access Control for Team Members"
    - **Details**: Implement the ability for team leads to assign roles (e.g., admin, editor, viewer) to new team members, defining their access levels to repositories and project settings.

Third User Story

3. As a project manager, I want to be able to track the progress of all active repositories across my team, so I can ensure that projects are moving forward efficiently and that no blockers are affecting the team.

Task:

- Build a dashboard for tracking repository activity.

Tickets:

- **Ticket 1 Title**: "Design Repository Activity Dashboard"
  - **Details**: Create a dashboard that provides an overview of active repositories. Include metrics such as recent commits, merge requests, and team activity over time.
- **Ticket 2 Title**: "Implement Repository Status Overview"
  - **Details**: Develop backend functionality to track repository statuses, such as open merge requests, pending reviews, and conflicts. Ensure the data is displayed in a way that is easy to understand for non-technical project managers.

Formal Requirements:

Goal:

Enable users to experience SnickerSync's unique sync and merge process, gathering feedback on usability, fun, and effectiveness, to determine whether the feature should be adopted more widely across GiggleGit.

Non-Goal:

Optimize the performance or scalability of SnickerSync for large repositories during the user study. (The focus is on evaluating user experience, not fine-tuning performance yet.)

Non-Functional Requirements:

1. Secure Access Control
Ensure that only authorized PMs and designated team members have access to configure and maintain the different SnickerSync concepts.

- **Functional Requirement 1.1**: Implement role-based access control (RBAC) where PMs can assign specific team members access to SnickerSync configurations.
- **Functional Requirement 1.2**: Provide a UI for PMs to manage access permissions and assign roles to other users within SnickerSync.

2. Random Assignment for User Studies
The user study must assign participants to control groups or variants of SnickerSync randomly to ensure unbiased results.

- **Functional Requirement 2.1**: Create an automated system that randomly assigns participants to control or experimental groups based on predefined study parameters (e.g., group size, role, project type).
- **Functional Requirement 2.2**: Ensure that user assignment to groups is tracked, allowing PMs to monitor which users were placed in control or experimental groups and their interactions with SnickerSync.