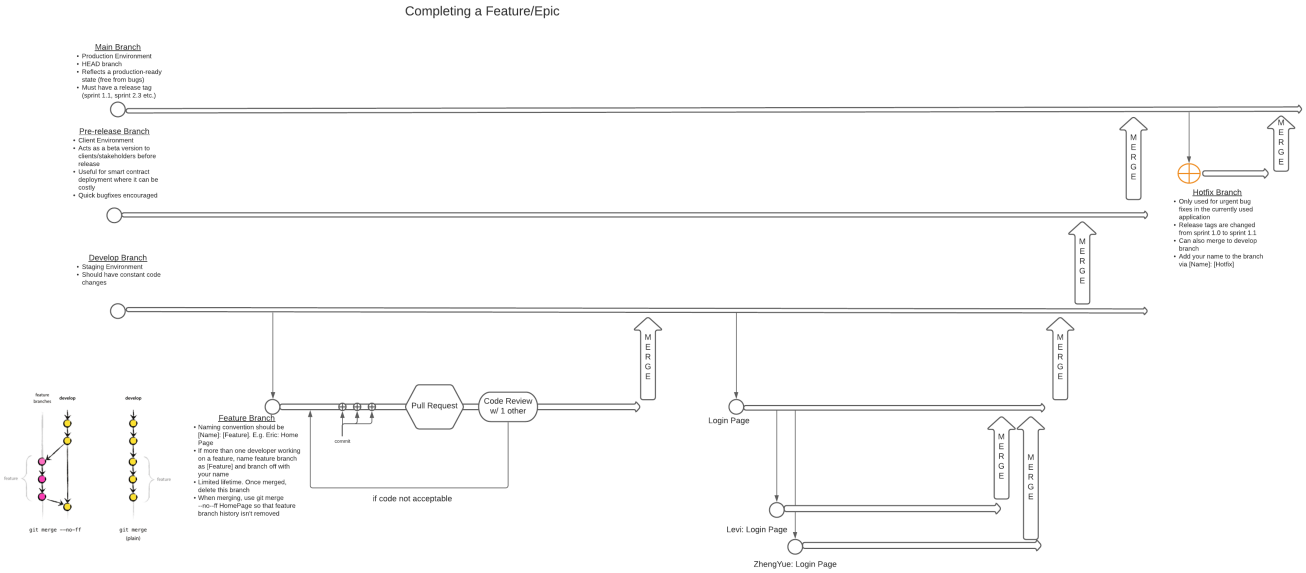
Git Workflow and Review Guidelines

Introduction

- A Git Workflow is a convention for how to use Git to progress with work consistently and effectively.
- Git workflows benefit projects as it:
    - Encourages team members to utilise Git consistently.
    - Ensure the team is aware of all other member's actions to the Git, promotes collaboration.
    - Clarifies branch responsibility which allows for easier parallel development.
    - Provides structure for Git and cleans it up.

Our end-to-end Git Process

Diagram 1: Git Process Workflow



Notes:

- Merging individual feature branches does not require a code review, however, one can be recommended at the team's discretion.
- Regression Testing should be performed regularly (after each merge?)
- Duplicate branches or extra branches are not recommended. Temporary branches can be made but should not be kept for long.

Code Review

Code Review Checklist:

- [ ] Am I able to understand the code easily?
- [ ] Is the code written following the coding standards/guidelines
- [ ] Is the same code duplicated more than twice?
- [ ] Can I unit test / debug the code easily to find the root cause?
- [ ] Is this function or class too big? If yes, is the function or class having too many responsibilities?

Coding Standards/Guidelines:

- We will be using Linter to aid in trivial coding conventions (tabspace, indents etc.)
- Proper naming convention = underlines (we can decide this)
- Code lines should not exceed 90 characters
- No hard coding, use constants as needed

- Group similar values under an enumeration
- Comment "NEEDS WORK" on areas that are still in progress
- Ensure the 'Open Closed Principle' where adding new functionality should be written in new classes and functions