GitFlow Workflow

GitFlow practices are used in companies to have an organized workflow when collaborating with a large number of people on a project. This helps prevent merge conflicts and allows all members of the team to understand the current state of a project.

For FlyUC, we will use a simple version of GitFlow standards to ensure that our work is organized long-term. Since we aren't building out a full software product the rules to follow will be very simple.

Good Resources:
https://blog.axosoft.com/gitflow/#:~:text=Workflow%20Model,of%20source%20code%20in%20production.
https://nvie.com/posts/a-successful-git-branching-model/

How will FlyUC apply GitFlow?

As our coding tasks become more collaborative, the GitFlow model will allows us to stay organized as a team. The resources above will go into more detail, however, some areas of the model we won't use as it won't be necessary.

Types of branches we will use:

- Feature branches
- Release branches (tags as well)
- Hotfix branches (bug fixes)

We won't use the develop - master relationship branch as we aren't producing a full length software project requiring multiple full length iterations.

Main branch: master

Essentially, master has the cleanest, most up-to-date development. When we want to grow our project, we will branch off of master using the 3 types of branches to make fixes or create features. After thorough testing, the independent changes made in those branches will be merged back into master. Once we have a version of master we are satisfied with, we create a release branch which shows the version number and allows us to cleanly test that state of the project.

When to use what branch?

- Feature branches - used to develop new features for the project's end goal
    - Branch off of **master**

    - Once approved (thoroughly tested) merge back into **master**
    - Branch naming convention: describes what the new feature will be
        - Ex: feature/submit-button-functionality
- Release branches - used to test a current verison of the project/end goal. Creating a branch helps debug problems and ensure a clean testing environment.
    - Branch off of **master**
    - **Doesn't merge back into any branch**
    - Tag last commit of the final version of the release
    - Naming convention: release number
        - Ex: release-0.0.1
- Hotfixes branches - fix bugs found in the master, or feature branches
    - Branch off of **master**
    - Merge back into **master**
    - Tag bug commits (Bug#__)
    - Naming convention: describe bug
        - Ex: hotfix/bug-1-fix-button-functionality