**Standard Operating Procedure (SOP): Git Workflow for WS Companion**

**Purpose**

To establish a clean, consistent, and scalable Git workflow for the WS Companion project, especially during the transition from the classical UI to the refactored UI design.

**Branch Structure**

**Permanent Branches**

* **main**: Production-ready, stable releases only
* **refactor-ui**: Long-lived base branch for the redesigned UI and architecture
* **classical-ui**: Archived branch preserving the original/classical version
* **development**: Staging area for merging feature branches and performing integration testing

**Feature Branches**

* Created from refactor-ui
* Naming format: feature/<feature-name>

Example:

git checkout refactor-ui

git checkout -b feature/productivity-dashboard

**Git Workflow Steps**

**1. Prepare Stable App**

* Tag current stable app:

git checkout main

git tag -a v1.8 -m "Stable version before UI refactor"

git push origin v1.8

* Create classical-ui:

git checkout -b classical-ui main

git push -u origin classical-ui

**2. Create Refactor Base**

git checkout -b refactor-ui main

# Begin redesign work

**3. Build Features on Refactor Base**

git checkout refactor-ui

git checkout -b feature/<feature-name>

**4. Merge Features into Development**

git checkout development

git merge feature/<feature-name>

**5. Final Merge to Main and Tag Release**

git checkout main

git merge development

git tag -a v2.0 -m "New UI release"

git push origin v2.0

**Tagging Releases**

Use annotated tags for versioning:

git tag -a v2.1 -m "Release notes here"

git push origin v2.1

**Notes**

* Avoid merging feature branches directly into main.
* Delete feature branches after merge to keep repo clean:

git branch -d feature/<name>

git push origin --delete feature/<name>

**Folder**

Save this file in:

strategy/git\_workflow\_sop.md