Developer + QA Process Guide

Bikini Bottom (STSWENG S21)

Initial Setup (One-Time Only)

1. Clone Repository

```
None
git clone
https://github.com/haniellejermayn/JJ-Apartments-Property-Management-System.git
cd JJ-Apartments-Property-Management-System
```

2. Set Up Security Hook

```
None
chmod +x .githooks/pre-commit
git config core.hooksPath .githooks
```

Note: *When you (accidentally) commit changes to* ***application.properties****, you will see a security notice. This is normal and the hook automatically resets the file, no need to reset it manually.*

Working on a Task

Step 1: Pick a Task from To Do (or Backlog)

Go to the Kanban Board and pick a task and **move the task to “In Progress”** on the Kanban Board.

Step 2: Create Feature Branch

```
None
# Get latest code
git checkout development
git pull origin development

# Create your branch based on the task
git checkout -b feat/view-subtenant-apartments-modal

# Branch naming examples:
# feat/view-subtenant-apartments-modal
# feat/view-subtenant-tenants-modal
# feat/api-fetch-unit-tenants
# feat/curr-occupants
```

Branch naming format: *feat/brief-task-description*

Step 3: Code & Commit

```
None
# Work on your task
# Test locally as you go

# Stage changes
git add .

# Commit with clear message referencing the task
git commit -m "feat(#41): add view button and modal in Apartments tab"

# Push to backup your work
git push origin feat/view-subtenant-apartments-modal
```

Step 4: Keep Your Branch Updated

Do this regularly, especially before creating PRs:

```
None
# Get latest changes from team
git checkout development
git pull origin development
```

```
# Merge into your branch
git checkout feat/view-subtenant-apartments-modal
git merge development
# If conflicts appear, fix them in VS Code, then:
git add .
git commit -m "Merge development into feat/view-subtenant-apartments-modal"
git push origin feat/view-subtenant-apartments-modal
```

Step 5: Move Task to “In Review”

When your task is complete:

1. Test locally (make sure it works!)
2. Commit and push final changes
3. Move task to **“In Review”** on Kanban Board
4. Create Pull Request (PR)

Step 6: Create Pull Request

On GitHub:

1. Go to repository → Pull requests → New pull request
2. Set: ***feat/view-subtenant-apartments-modal → development***
3. No need to fill out description
4. Click **Create pull request**

Step 7: Wait for CI

Watch your PR for:

- **CI Running** (Yellow dot) - Wait...
- **CI Passed** (Green checkmark) - Merge
- **CI Failed** (Red X) - Fix errors

Step 8a: If CI Passes → Merge

1. Click **Merge pull request**

2. Click **Confirm merge**
3. Discord should be automatically notified by webhook
4. Done!

Step 8b: If CI Fails → Fix Errors

1. Click **Details** on the failed check
2. Read the error message
    - If it's a linting error, run ***npm run lint*** to see the issues.
3. Fix the error locally
4. Commit and push

```
None
git add .
git commit -m "fix: Resolve linting error"
git push origin feat/view-subtenant-apartments-modal
```

5. CI will automatically re-run

Step 9: Clean Up

```
None
# Switch back to development
git checkout development

# Get latest (including your merged code)
git pull origin development
# Delete old branch
git branch -d feat/view-subtenant-apartments-modal

# Pick next task from to do or backlog!
```

When ALL Tasks in a Feature are "In Review"

Example: SUB-TENANT-1.1 (View Sub-tenants)

All tasks completed:

✔ Task #1: View button in Apartments tab (merged)
✔ Task #2: View button in Tenants tab (merged)
✔ Task #3: API to fetch tenants (merged)
✔ Task #5: Update unit backend (merged)
✔ Task #4: Test script prepared (QA)

Now, **ping @QA in #qa-testing!**

Fixing Bugs Reported by QA

Step 1: QA Tests the feature in development branch

- QA tests directly in ***development***
- QA creates **BUG issues** on the Kanban Board with **To Do status**
- QA pings devs in **#qa-testing**
    - Just let the devs know that the feature has been tested and let them know if there are any bugs then direct them to the Kanban Board and/or test script document.

Step 2: Move BUG to “In Progress” & Create Fix Branch

- Before starting fixing the bug, make sure to move the issue to **In Progress**

```
None
# Start from development
git checkout development
git pull origin development

# Create fix branch
git checkout -b fix/subtenant-modal-undefined-message
```

Step 3: Fix the Bug

```
None
# Reproduce bug locally
# Find and fix the issue
# Test with units that have NO sub-tenants

git add .
git commit -m "fix(#60): show proper message when no sub-tenants exist"
git push origin fix/subtenant-modal-undefined-message
```

Step 4: Move BUG to “In Review” & Create PR for Fix

```
None
Title: Fix undefined message in sub-tenant modal (#60)
Changes:
- Fixed conditional rendering in modal
- Now shows "No sub-tenants currently residing in this unit"
- Added null/undefined checks
```

Step 5: Merge and Notify QA

- Just ping **@QA in #qa-testing** again to update them that the bug has been resolved
- If the bug was resolved, **QA will move the issue to Done**, automatically closing the issue.

That’s it! The process just repeats itself for each feature each increment.