EcomiqX – Git Collaboration & Release Workflow

This document outlines the robust Git collaboration and release workflow for EcomiqX, a cutting-edge e-commerce solution developed by Tecresearch. Our primary goal is to enable multiple developers to work on a shared codebase efficiently, preventing production disruptions and accidental overwrites.

This workflow ensures a stable, scalable, and secure development environment, crucial for enterprise-level applications.

The Team and Repository

- **Repository:** https://github.com/tecresearch/ecomiqx
- **Goal:** To allow multiple developers to work on one codebase without breaking production and without overwriting each other’ s work.

At Tecresearch, EcomiqX represents our commitment to innovative e-commerce solutions. A well-defined Git strategy is paramount to achieving our development objectives.

1	TEAM ROLE	**Brijesh** Tech Lead & Architect
2	TEAM ROLE	**Shrisht** Developer

Defining the Base Code

The base code serves as the immutable foundation of our EcomiqX project. Its stability is paramount to the integrity of our production environment.

Included in Base Code:

- /frontend skeleton
- /backend Spring Boot skeleton
- Database configuration
- Project structure
- Core configuration

Excluded from Base Code:

- Login logic
- Product management features
- Order processing modules
- Payment gateway integrations
- UI pages (beyond skeleton)

Only Brijesh, as the Tech Lead, is authorised to make direct changes to the base code, ensuring controlled evolution of our core platform.

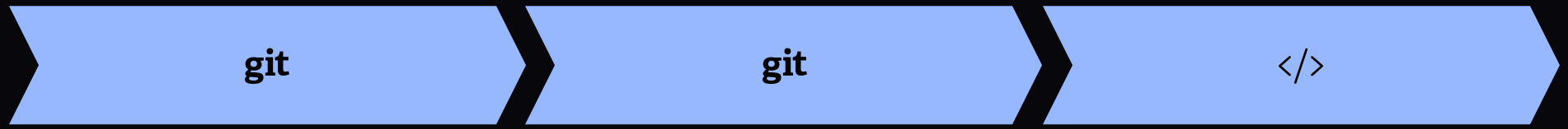
Our GitFlow Branch Strategy

EcomiqX rigorously adheres to the GitFlow branching model to ensure a structured and predictable development lifecycle. Each branch serves a specific purpose, preventing conflicts and maintaining code quality.

main	Live production code
develop	Integration and testing environment
feature-*	Dedicated for new feature development
release/*	Preparation for production releases
hotfix/*	Urgent fixes for production issues

Project Initialization: Establishing the Foundation

The EcomiqX project begins with a careful setup of the core codebase, ensuring a stable starting point for all subsequent development. This initial phase defines the structure upon which all features will be built.



Initial Push to Main

Base code pushed to the `main` branch.

Develop Branch Creation

`develop` created as an exact copy of `main`.

Branch State

`main` and `develop` are synchronised (v0.1).

This foundational step ensures that `main` represents a stable, shippable version of the platform, while `develop` becomes the active working branch for ongoing enhancements.

Daily Development Flow: Feature Branching

Both Brijesh and Shrisht adhere to a consistent daily development process, centred around isolated feature branches. This approach minimises conflicts and promotes clean code integration.

Checkout Develop

Always start by checking out the latest `develop` branch.

Pull Latest Changes

Ensure your local `develop` is up-to-date with `git pull`.

Create Feature Branch

Branch off `develop` for new features (e.g., `feature-login`).

Work & Commit

Develop your feature, commit changes with descriptive messages, and push to origin.

Create Pull Request

Submit a Pull Request targeting `develop` for review and approval.

Crucially, merges into `develop` only occur after thorough Pull Request approval, maintaining code quality and integrity.

Feature Integration: Merging into Develop

All newly developed features are systematically merged into the `develop` branch. This strict policy ensures that `develop` always reflects the most current, integrated version of the EcomiqX application, ready for comprehensive testing.

Feature-to-Develop

All `feature-*` branches are **always** merged into `develop`.

Never to Main

Features are **never** directly merged into the `main` branch.

This clear separation prevents unfinished or untested code from inadvertently reaching the production branch, safeguarding the stability of our live EcomiqX environment.

Release to Production: A Controlled Deployment

The process for releasing EcomiqX features to production is highly structured, ensuring stability and a smooth transition from development to live environments. This multi-stage approach includes dedicated QA and careful merging.

Create Release Branch

When `develop` is ready, a `release/X.Y.Z` branch is created.

QA Testing

Extensive Quality Assurance testing is performed on the `release` branch.

Merge to Main

If stable, merge `release/X.Y.Z` into `main` and push.

Merge to Develop

Also merge `release/X.Y.Z` into `develop` to incorporate release fixes.

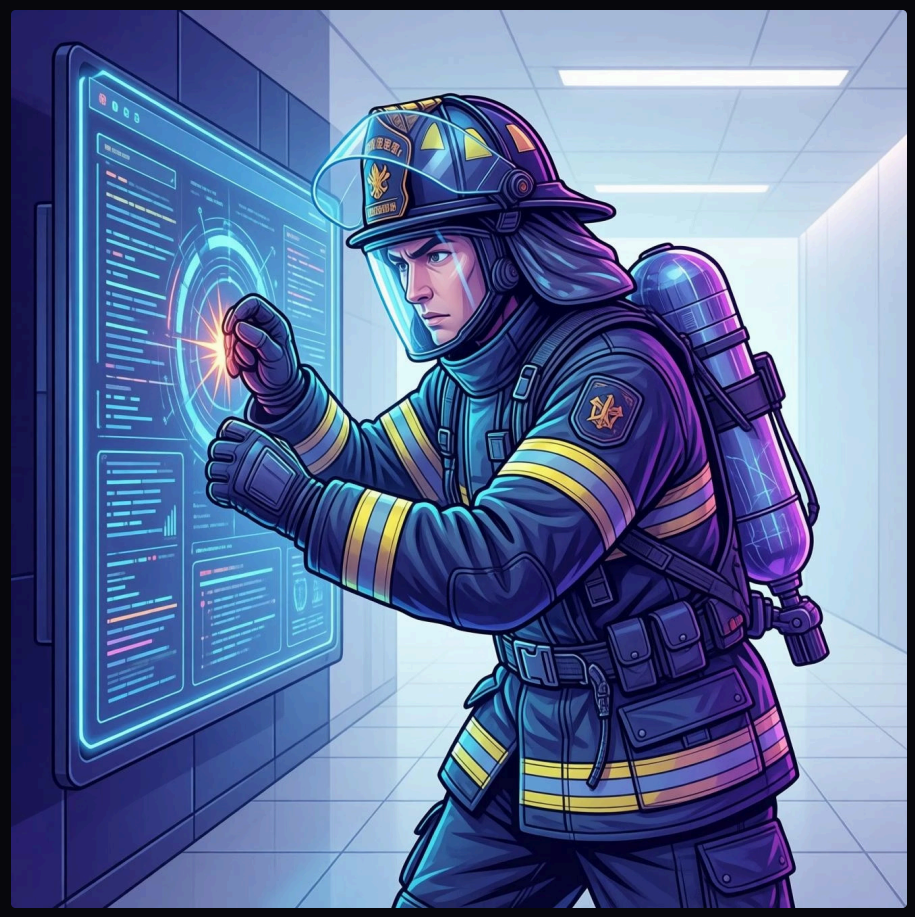
Clean Up

Delete the `release` branch after successful deployment.

This meticulous release strategy ensures that `main` always reflects the current live production version, while `develop` prepares for future updates.

Hotfix Protocol: Addressing Production Bugs

In the event of a critical issue on the EcomiqX production environment, a rapid and controlled hotfix process is initiated. This protocol prioritises immediate resolution while maintaining the integrity of our branching strategy.



Checkout Main

Start by checking out the `main` branch.

Create Hotfix Branch

Branch off `main` for the fix (e.g., `hotfix-payment`).

Fix and Commit

Implement the fix, commit changes.

Merge into Main

Merge the hotfix back into `main`.

Merge into Develop

Also merge the hotfix into `develop` to prevent regression.

This dual merge ensures that critical fixes are immediately deployed to production and are also integrated into the ongoing development cycle.

Robust Branch Protection & Ownership

To prevent accidental damage and enforce best practices, critical branches are protected with stringent rules. This, combined with clear ownership, forms the backbone of EcomiqX's secure development environment.

2

Protected Branches

Main and Develop branches are protected.

3

Protection Rules

- Pull Request required
- Review required
- No direct push

Clear ownership defines responsibilities:

> **Brijesh:** Base code, architecture, release & production.
>
> **Shrisht:** Feature development.

Shrisht cannot modify base configuration without Pull Request approval, fostering a culture of peer review and shared responsibility. This system culminates in a safe, collaborative, and enterprise-level Git workflow for EcomiqX.