# SAMFMS Git Branching Strategy Document

## Purpose

This document defines the **Git branching strategy** for the **SAMFMS (Scalable and Modular Fleet Management System)** project. It outlines how branches should be created, named, used, and merged, ensuring a clean, scalable, and collaborative development environment that supports modular development and CI/CD pipelines.

---

## Branch Types Overview

| **Branch** | **Purpose** | **Naming Convention** |
|---|---|---|
| main | Stable production-ready code. CI/CD deploys from here. | main |
| develop | Integration branch for features; base for release branches. | develop |
| feature/* | New features or SBlocks. | feature/gps-sblock, feature/dashboard-ui |
| bugfix/* | Minor fixes from testing or QA. | bugfix/driver-ui-crash |
| hotfix/* | Critical fixes for main. | hotfix/service-export-bug |
| release/* | Pre-release stabilization. | release/v1.0.0 |
| experiment/* | Experimental or prototype branches. | experiment/ai-routing |

---

## Branching Flow

```
graph TD;
 A[main] -->|branch| B[develop]
 B -->|branch| C[feature/smart-trip-planning]
 C -->|merge PR| B
 B -->|branch| D[release/v1.0.0]
 D -->|merge PR| A
 B -->|branch| E[bugfix/ui-glitch]
 E -->|merge PR| B
```

A -->|branch| F[hotfix/map-crash]

F -->|merge PR| A

---

**Commit Message Format**

**To ensure clarity and consistency, all commit messages should follow the Conventional Commits format:**

**<type>(optional-scope): short summary**

**[optional body]**

**[optional footer]**

**Types**

| **Type** | **Description** |
|---|---|
| **feat** | **A new feature (e.g., a new SBlock, dashboard widget)** |
| **fix** | **A bug fix** |
| **docs** | **Documentation changes** |
| **style** | **Code style changes (formatting, whitespace – no logic changes)** |
| **refactor** | **Code changes that neither fix a bug nor add a feature** |
| **perf** | **Performance improvement** |
| **test** | **Adding or modifying tests** |
| **chore** | **Maintenance tasks (build scripts, dependencies, etc.)** |
| **ci** | **Changes to CI/CD configuration** |

🔹 **Examples**

**bash**

**Copy code**

**feat(gps-sblock): add support for displaying vehicle routes on dashboard**

**fix(trip-planner): resolve crash when generating route with invalid address**

**docs(mcore): update README with plugin loading instructions**

**style(ui): adjust padding for dashboard cards**

**refactor(mcore): extract plugin interface into separate module**

**test(maintenance-sblock): add tests for service reminder logic**

**chore: update project dependencies**

🔹 **Rules**

- **Use imperative mood (e.g., “add” not “added”).**
- **Keep the subject line under 72 characters.**
- **Separate body from subject with a blank line (if body is present).**
- **Use the body to explain why the change was made (if necessary).**
- **Use the footer to reference Jira tickets, issues, or breaking changes.**

---

**Rules & Conventions**

**main**

- **Deployable at all times**.
- Only maintainers merge into main via Pull Requests.
- Merge only from release/* or hotfix/*.

**develop**

- **Main integration branch** for features.
- All feature/*, bugfix/*, and experiment/* branches merge here.
- Merge via **Pull Request** with code review.

**feature/***

- Branch from: develop
- Merge to: develop
- One feature per branch (e.g., feature/vehicle-maintenance-sblock)
- Recommended naming format:
    - feature/<component>-<short-description>

**bugfix/***

- For small issues found during testing or QA.
- Merge into develop.

**hotfix/***

- Urgent fixes to production code.
- Merge into both main and develop.

**release/***

- Used for **QA, documentation, and final testing**.
- Versioned (e.g., release/v1.0.0).
- Only small fixes and polish allowed.

**experiment/***

- Used for experimental modules or ideas.
- May be discarded or merged depending on review.
- E.g., experiment/woocommerce-integration

---

**SBlock Workflow Example**

Suppose the team is working on the **Trip Planning SBlock**:

1. Create branch: feature/trip-planning-sblock
2. Push all modular code inside /sblocks/trip-planning/
3. When ready, open a PR to develop and request a review.
4. Upon merge, it becomes part of the develop build.

---

**Pull Request Guidelines**

- One PR per feature/bug/patch.
- PR title should follow format: [Feature] Trip Planning SBlock
- Description must include:
    - Summary of changes
    - Related issue/ticket
    - How to test the changes
- Requires at least **one review** and **CI pipeline pass**.

---

**CI/CD Integration (Optional)**

- **Trigger Build/Test** on every PR to develop or main.
- **Auto Deploy** from main to production server.
- **Auto Tag Release** when merging release/* to main.

---

**Suggested Team Roles (For Branch Ownership)**

| **Role** | **Branch Responsibility** |
|---|---|
| UI Engineer | feature/dashboard-ui, bugfix/ui-* |
| Backend Engineer | feature/mcore-comm, feature/api-* |
| DevOps | hotfix/*, release/*, CI/CD |
| AI Engineer | experiment/ai-routing, feature/fleet-analytics |
| Integration Lead | develop, code reviews |

---

**Notes**

- Branches should be **short-lived** (merged within a week if possible).
- Delete branches after merge to reduce clutter.
- Each SBlock should live in its own folder/module structure inside the repo.