🚀 SOP GIT - Pengembangan Proyek

🌿 Git Branching

Setiap repository akan memiliki **3 branch utama**:

- `master`
- `development`
- `production`

Setiap pembuatan branch baru, buat branch dengan **base** `master` .

Gunakan format: `<tipe>/<judul>`

📌 Tipe Branch:

- `story` → Untuk fitur atau use case baru
- `task` → Untuk bug fixing, performance improvement, refactor, dll

📝 Format Judul:

Gunakan **kebab-case**

✅ Contoh:

- `story/api-attendance`
- `story/page-attendance`
- `task/improve-sql-performance-on-xxxx-method`

Setelah selesai:

> *Buat **Pull Request ke** `master`*

🎨 Code Styling & Repository

Mohon perhatikan hal-hal berikut:

1. Gunakan penamaan variabel, fungsi, dan kelas **yang bermakna**
2. Penyingkatan harus **mudah ditebak & terbaca**

   Misalkan, codeStylingAndRepository, terlalu panjang, disingkat menjadi:

   - ✅ `codeStyleNRepo`
   - ❌ `csnr` , `cdStNrep`
3. Penamaan:
   - 🏷️ Kelas: `PascalCase` → `ClassName`
   - 🏷️ Fungsi & Variabel: `camelCase` → `fungsiDanVariabel`
   - 🗂️ Folder: `snake_case` → `folder_styling`
   - 📄 File: `kebab-case` → `file-styling.tsx`
   - ⚛️ Komponen React & File: `PascalCase` → `NamaKomponen`

📝 Semantic Commit Message

Gunakan format semantic untuk commit message:

📜 Format:

```
<type>(<scope>): <short description>
```

📦 Tipe Commit:

Type	Deskripsi
`feat`	✨ Fitur baru untuk user
`fix`	🦎 Bug fix untuk user
`docs`	📚 Perubahan dokumentasi
`style`	🎨 Format kode (spasi, titik koma, dll) tanpa ubah logic produksi
`refactor`	🔨 Refactor kode produksi (tanpa mengubah fungsionalitas), eg. renaming a variable

Type	Deskripsi
`test`	💉 Menambahkan atau mengubah test
`chore`	🔧 Update hal-hal non-produksi (config, task runner, dll)