1.Mục tiêu

- Tránh đè code lên nhau.
- Kiểm soát version qua từng sprint.
- Code luôn được review trước khi merge.
- Hạn chế tối đa lỗi lên main.

2.Branch Strategy

Có 3 loại branch chính:

main

- Code sau cùng của mỗi sprint.
- Chứa code ổn định, đã test.
- **không ai được tự ý merge vào main**

develop

- Nơi tập hợp code từ tất cả feature.
- Sau khi kết thúc sprint merge develop → main.

feature/*

- Dùng cho từng task cụ thể.
- Mỗi task = 1 branch dạng:
    - `feature/login-ui`
    - `feature/register-api`
    - `feature/chat-socket`
    - `feature/fix-bug-message-scroll`
- Khi hoàn thành → tạo Pull Request → merge vào develop.

mọi người lưu ý là luôn phải tạo branch feature từ develop trước khi code và pull request khi merge vào develop. (sẽ không thể push nếu code trực tiếp trên develop). Không ai được tự merge code vào main

3. Git Flow chi tiết

- **bắt đầu một task (hoặc một nhóm task liên quan)**
  - **git checkout develop** (nếu đang ở main thì chuyển sang develop)
  - **git pull** (nhớ kéo code mới nhất về)
  - **git checkout -b feature/<ten-task>** (nhớ tạo branch mới từ develop trước khi code)
- **trong lúc làm (trong feature)**
  - **git add .**
  - **git commit -m "feat: mô tả ngắn gọn thay đổi"**
  - **git push -u origin feature/<ten-task>**
- **kết thúc task mở pull request**
  - **mở PR: `feature/<ten-task>` → `develop`**
  - **`git branch -d feature/<ten-task>`** (`Xoá branch sau merge`)

4. Quy tắc đặt tên branch & commit

- **Tên branch**
  - feature/… ví dụ: feature/chat-ui
- **Quy tắc commit**
  - `feat:` – thêm tính năng
  - `fix:` – sửa lỗi
  - `docs:` – chỉnh tài liệu
  - `style:` – format code
  - `refactor:` – viết lại code nhưng không thay đổi tính năng
  - `chore:` – config, setup tool

Ví dụ commit:

- feat: tạo api login cho user
- fix: sửa lỗi không load được danh sách chat
- refactor: tách component MessageItem