TỔNG KẾT CẤU TRÚC VÀ LUỒNG CHẠY PIPELINE V5 LAZY

# 1. Sơ đồ thư mục chính

timeline\_rack/  
├── modules/  
│ └── marker\_utils.py # Xử lý marker từ file timeline robot  
├── timeline\_rack\_v5\_lazy.py # Hàm chính xử lý pipeline lazy rack  
├── request\_cc\_from\_robot\_pipeline.py # Gọi pipeline robot và nạp dữ liệu CC khi cần  
├── run\_rack\_lazy.bat # (Tùy chọn) Script khởi động chạy pipeline lazy  
└── (các tệp log đầu ra nếu cần)

# 2. Tổng kết cấu trúc pipeline V5 Lazy

Hàm trung tâm: timeline\_rack\_v5\_lazy(request\_cc\_fn)

Mục tiêu: chỉ nạp CC khi cần (lazy loading), xử lý từng vị trí rack một cách chính xác và tiết kiệm tài nguyên.

## Các bước xử lý chính:

1. 1. Gọi request\_cc\_fn(1) để lấy dữ liệu CC1 từ pipeline robot.
2. 2. Xác định first\_pos và last\_pos bằng find\_first\_last\_position\_from\_marker(df\_cc1).
3. 3. Tạo action\_db cho CC1 bằng build\_action\_db() và lưu vào cc\_pool.
4. 4. Sinh danh sách các vị trí rack đi qua bằng extract\_positions().
5. 5. Duyệt từng vị trí pos trong positions:
6. - Gọi process\_position(pos, cc\_pool, current\_cc, ...)
7. - Nếu thiếu CC tiếp theo, tạm dừng xử lý pos hiện tại, gọi request\_cc\_fn(cc+1), rồi tiếp tục lại chính pos đó.
8. - Nếu đủ dữ liệu, thêm kết quả vào timeline\_result.
9. 6. Trả về toàn bộ timeline\_result dưới dạng DataFrame.

## Lưu ý quan trọng:

* ⚠️ Mỗi pos có thể cần dữ liệu từ nhiều CC (VD: T\_in từ CCi, T\_out từ CCi+1).
* ⚠️ Không được bỏ qua pos nếu chưa có đủ dữ liệu.
* ✅ Sử dụng vòng lặp while để kiểm soát việc xử lý lại pos nếu cần sinh thêm CC.
* ✅ Chỉ tăng chỉ số i nếu pos đã được xử lý xong.