IOS TUTORIALS

TỐI ƯU HIỆU SUẤT UITABLEVIEW / UICOLLECTIONVIEW BẰNG CELL PREFETCHING & LAZY IMAGE LOADING

nguyen thi nhung

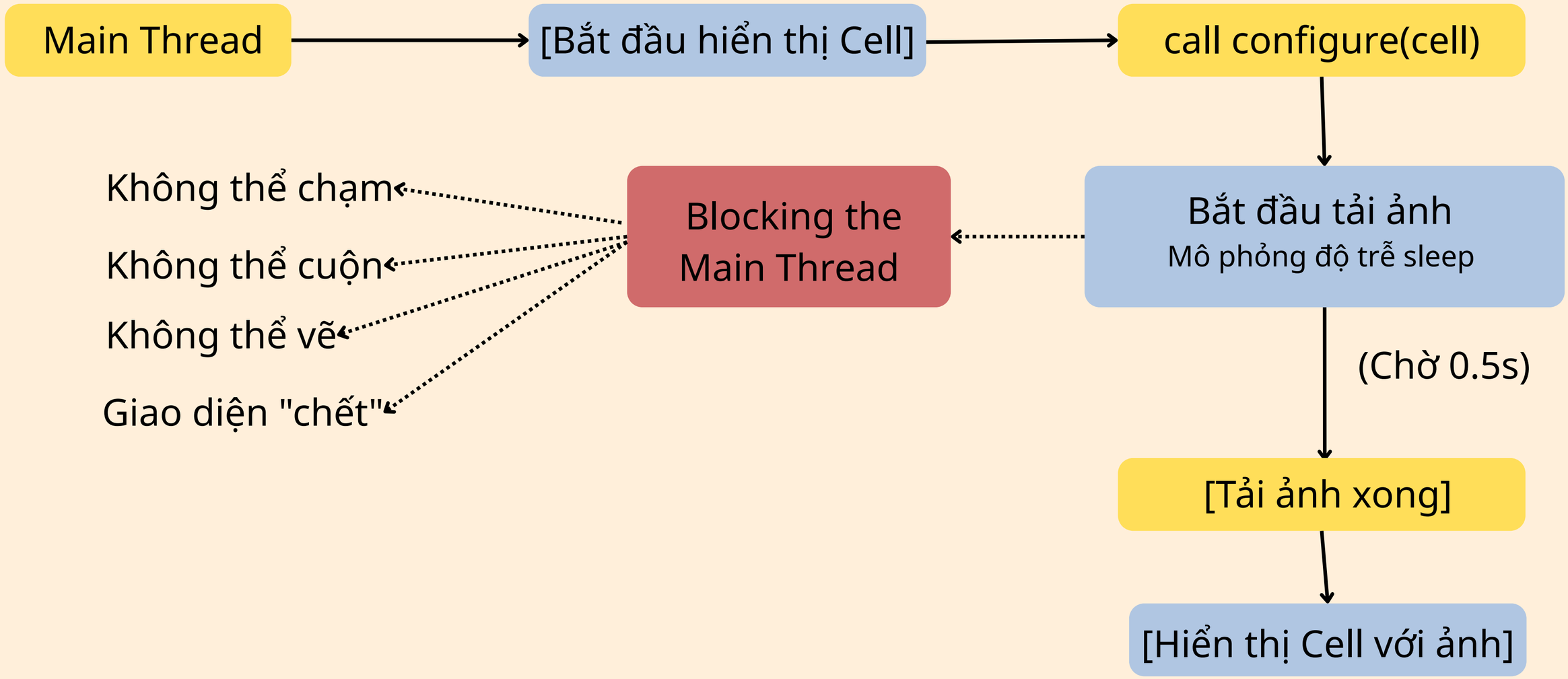
VẤN ĐỀ: TRẢI NHIỆM GIẬT LAG
TÌNH HUỐNG: HIỂN THỊ MỘT DANH SÁCH ẢNH

- Khi hiển thị một danh sách, chúng ta thường phải tải ảnh từ URL => Đây là một tác vụ có độ trễ không thể đoán trước được
- Mô phỏng bằng cách dùng Thread.sleep() để tượng trưng cho thời gian chờ ảnh từ mạng

```
public func configure(with model: Model) {
    self.myLabel.text = model.text
    Thread.sleep(forTimeInterval: 0.5)
    self.myImageView.image = UIImage(named: model.imageName)
}}
```

- Giao diện bị đóng băng, giật, lag khi scroll do kiến trúc ban đầu chưa được thiết kế để xử lý độ trễ

TẠI SAO GIAO DIỆN BỊ ĐÓNG BĂNG? BLOCKING THE MAIN THREAD



CELL PREFETCHING

GIẢI PHÁP “LÀM VIỆC TRƯỚC ĐỂ KHÔNG PHẢI CHỜ ĐỢI”

Nếu biết người dùng sắp cuộn đến đâu=> có thể tải ảnh trước.
Khi cell cần hiển thị, ảnh đã có sẵn trong cache!
Từ bị động chờ đợi sang chủ động chuẩn bị.

Nhờ giao thức **UITableViewDataSourcePrefetching => dự đoán các dòng sắp xuất hiện**

```
func tableView(_ tableView: UITableView, prefetchRowsAt indexPaths: [IndexPath]) {
    print("Prefetching rows: \(indexPaths.map { $0.row })")
    for indexPath in indexPaths {
        let model = models[indexPath.row]
        ImageManager.shared.preloadImage(named: model.imageName)
    }
}
```

CELL PREFETCHING
GIẢI PHÁP “LÀM VIỆC TRƯỚC ĐỂ KHÔNG PHẢI CHỜ ĐỢI”

lưu ảnh vào cache, trả về ảnh đã tải

lấy ảnh từ cache, không thì tải

tiền tải hình ảnh

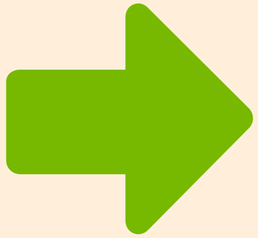
```
class ImageManager {
    static let shared = ImageManager()
    private let imageCache = NSCache<NSString, UIImage>()

    private init() {}
    private func loadImageSynchronously(named imageName: String) -> UIImage? {
        Thread.sleep(forTimeInterval: 0.5)
        let image = UIImage(named: imageName)
        if let image = image {
            self.imageCache.setObject(image, forKey: imageName as NSString)
        }
        return image
    }
    func getImage(named imageName: String) -> UIImage? {
        if let cachedImage = imageCache.object(forKey: imageName as NSString) {
            return cachedImage
        }
        return loadImageSynchronously(named: imageName)
    }
    func preloadImage(named imageName: String) {
        if imageCache.object(forKey: imageName as NSString) != nil {
            return
        }
        _ = loadImageSynchronously(named: imageName)
    }
}
```

Cuộn chậm rất mượt!

CELL PREFETCHING

GIẢI PHÁP “LÀM VIỆC TRƯỚC ĐỂ KHÔNG PHẢI CHỜ ĐỢI”

Khi người dùng vuốt mạnh, scroll nhanh,
prefetchRowsAt không có đủ thời gian để chạy trước.



Phải chuẩn bị cho cả kịch bản tồi tệ nhất
ImageManager (đồng bộ) tiến hóa thành ImageLoader (bất đồng bộ).

KẾT HỢP PREFETCHING VÀ LAZY LOADING
SỰ THAY ĐỔI CỐT LÕI: TỪ ĐỒNG BỘ SANG BẤT ĐỒNG BỘ.

Nếu ảnh chưa có trong cache khi configure được gọi, sẽ:

- Hiển thị một placeholder ngay lập tức.
- Không chặn luồng chính, mà yêu cầu ImageLoader tải ảnh trên luồng nền.
- Khi nào tải xong thì cập nhật lại UI.

```
class ImageLoader {
    static let shared = ImageLoader()
    private let imageCache = NSCache<NSString, UIImage>()

    private init() {}

    func loadImage(named imageName: String, completion: @escaping (UIImage?) -> Void) {
        if let cachedImage = imageCache.object(forKey: imageName as NSString) {
            completion(cachedImage)
            return
        }

        DispatchQueue.global(qos: .userInitiated).async {
            Thread.sleep(forTimeInterval: 0.5)
            let image = UIImage(named: imageName)

            if let image = image {
                self.imageCache.setObject(image, forKey: imageName as NSString)
            }

            DispatchQueue.main.async {
                completion(image)
            }
        }
    }

    func preloadImage(named imageName: String) {
        loadImage(named: imageName) { _ in }
    }
}
```

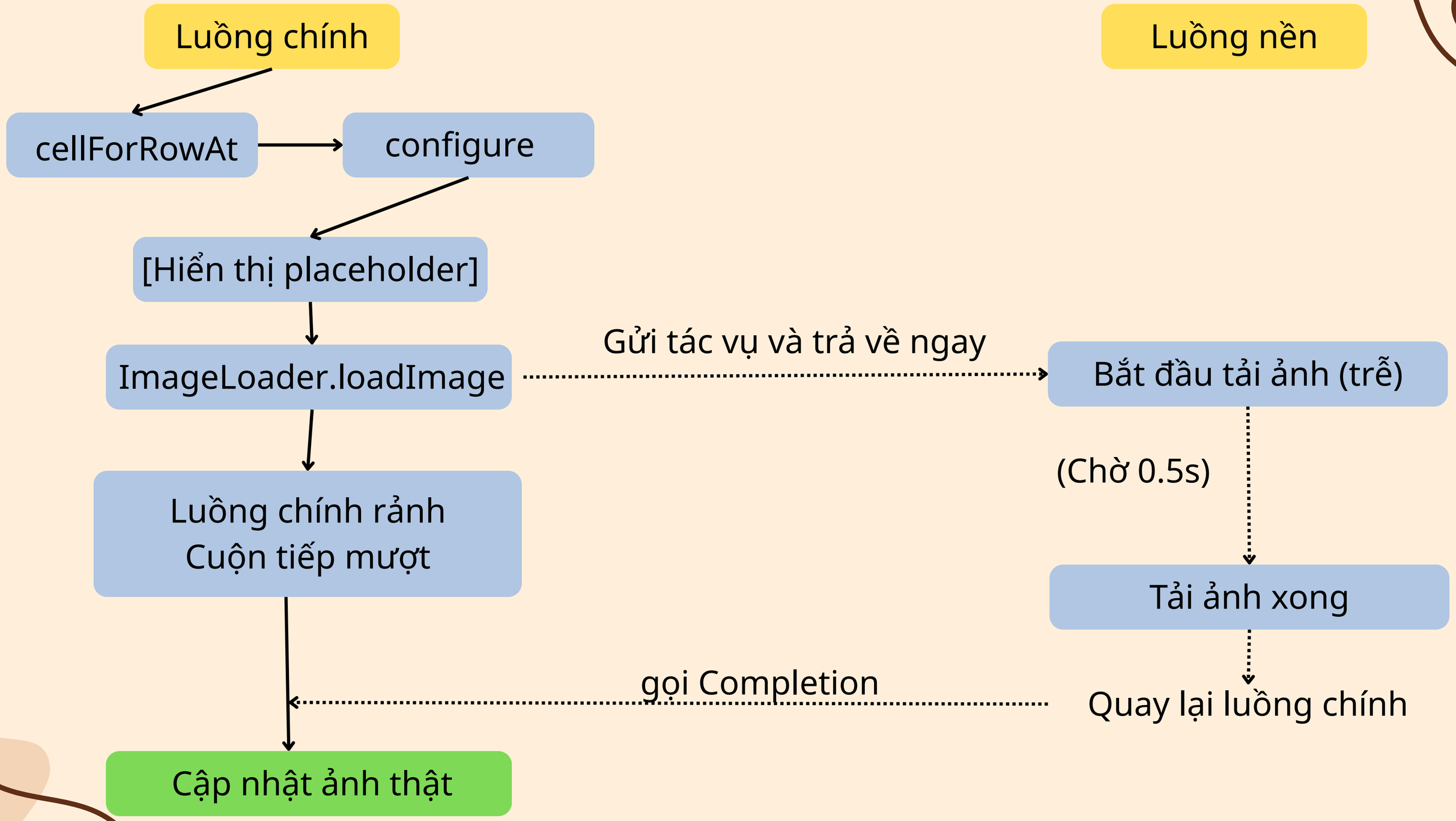
nếu có ảnh trong cache thì trả về sau khi tải

Tải ảnh bất đồng bộ lên luồng nền => complettion trên luồng chính

KẾT HỢP PREFETCHING VÀ LAZY LOADING
SỰ THAY ĐỔI CỐT LÕI: TỪ ĐỒNG BỘ SANG BẤT ĐỒNG BỘ.

```
    override func awakeFromNib() {
        super.awakeFromNib()
        myImageView.image = nil
        myImageView.backgroundColor = .white
    }
    override func prepareForReuse() {
        super.prepareForReuse()
        myLabel.text = nil
        myImageView.image = nil
        myImageView.backgroundColor = .white
    }

    public func configure(with model: Model) {
        self.myLabel.text = model.text
        self.myImageView.image = nil
        self.myImageView.backgroundColor = .white

        ImageLoader.shared.loadImage(named: model.imageName) { [weak self] image in
            self?.myImageView.image = image
        }
    }
}
```

=>Đặt placeholder


Luồng chính
Luồng nền
cellForRowAt
configure
[Hiển thị placeholder]
ImageLoader.loadImage
Gửi tác vụ và trả về ngay
Bắt đầu tải ảnh (trễ)
(Chờ 0.5s)
Luồng chính rảnh
Cuộn tiếp mượt
Tải ảnh xong
gọi Completion
Quay lại luồng chính
Cập nhật ảnh thật

SO SÁNH

	CƠ BẢN	CELL PREFTCHING	PREFETCHING + LAZY LOADING
TƯ DUY	Làm cho nó chạy	Làm việc trước	Chuẩn bị mọi thứ
LOGIC TẢI ẢNH	Dồng bộ	Đồng bộ	Bất đồng bộ
XỬ LÝ SCROLL NHANH	Đóng băng	Đóng băng	Mượt mà
ĐIỂM MẠNH	Đơn giản	Mượt khi scroll chậm	Mạnh mẽ và an toàn trong mọi trường hợp
ĐIỂM YẾU	Trải nghiệm tệ	Không an toàn	Không có điểm yếu rõ ràng

THANK YOU