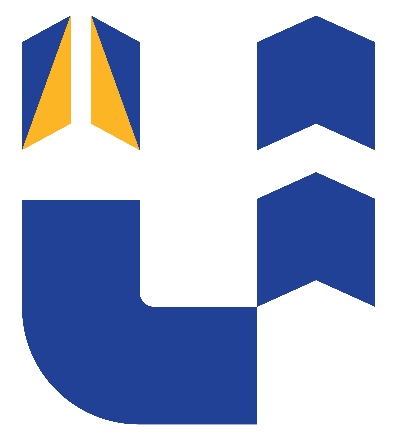
****

**LAB REPORT**

**Môn Học: Công nghệ mới**

**2025-2026**

Họ tên**:** Nguyễn Hữu Sang

**ID:** 22669281

**Lớp:** DHKTPM18C

**Khóa học:** 2025-2026

**Mục lục**

[CHƯƠNG 1: YÊU CẦU BỔ SUNG Lab3 (ADVANCED EXTENSION – TABLES & FEATURES) 3](#_Toc221282172)

[1.1 Link github 3](#_Toc221282173)

[1.2 Màn hình chạy thiết bị 3](#_Toc221282174)

[1.2.1 Màn hình đăng nhập 3](#_Toc221282175)

[1.2.2 Nhân viên 4](#_Toc221282176)

[1.2.3 Quản lý 5](#_Toc221282177)

[1.3 Sơ đồ quan hệ giữa các bảng DynamoDB 8](#_Toc221282178)

[1.4 Luồng xử lý: 10](#_Toc221282179)

[1.4.1 đăng nhập 10](#_Toc221282180)

[1.4.2 thêm sản phẩm 11](#_Toc221282181)

[1.5 So sánh: 11](#_Toc221282182)

[1.5.1 DynamoDB vs MySQL cho bài toán này 11](#_Toc221282183)

[1.6 Nhận xét ưu / nhược điểm mô hình NoSQL 13](#_Toc221282184)

[1.6.1 ƯU ĐIỂM: 13](#_Toc221282185)

[1.6.2 NHƯỢC ĐIỂM: 13](#_Toc221282186)

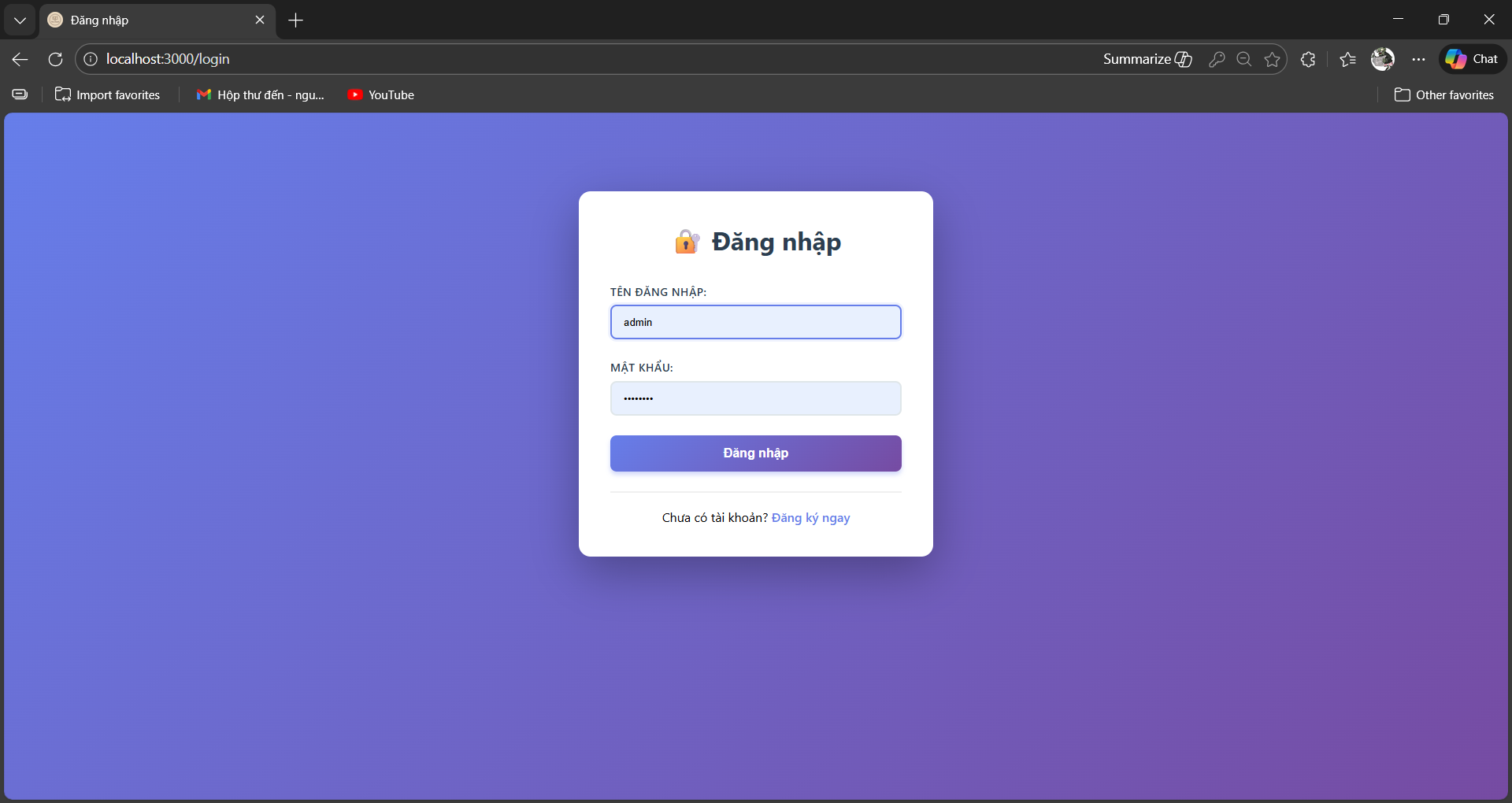
# YÊU CẦU BỔ SUNG Lab3 (ADVANCED EXTENSION – TABLES & FEATURES)

## Link github

<https://github.com/NguyenHuuSang04/CNM/tree/main/Lab_4>

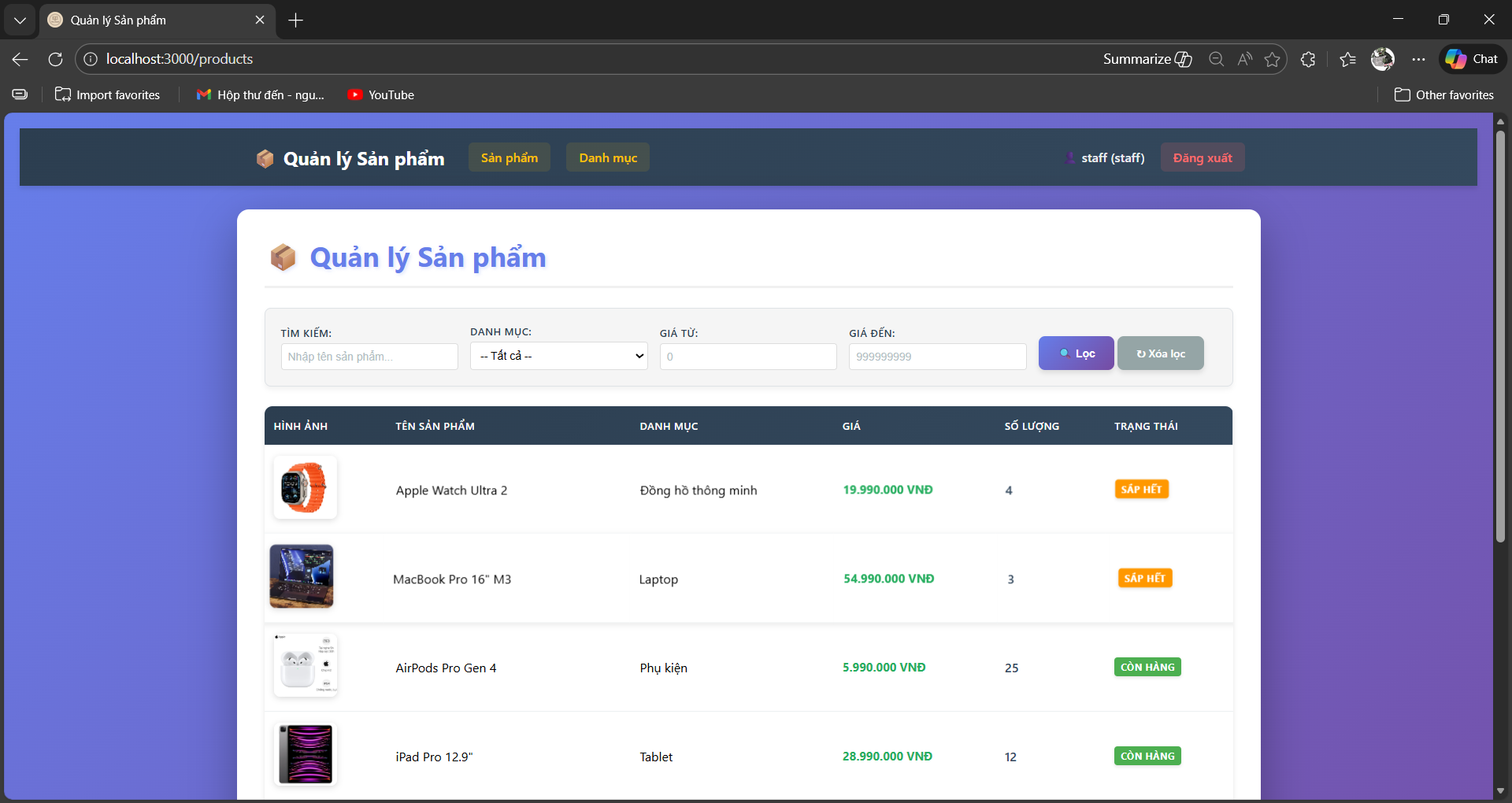
## Màn hình chạy thiết bị

### Màn hình đăng nhập

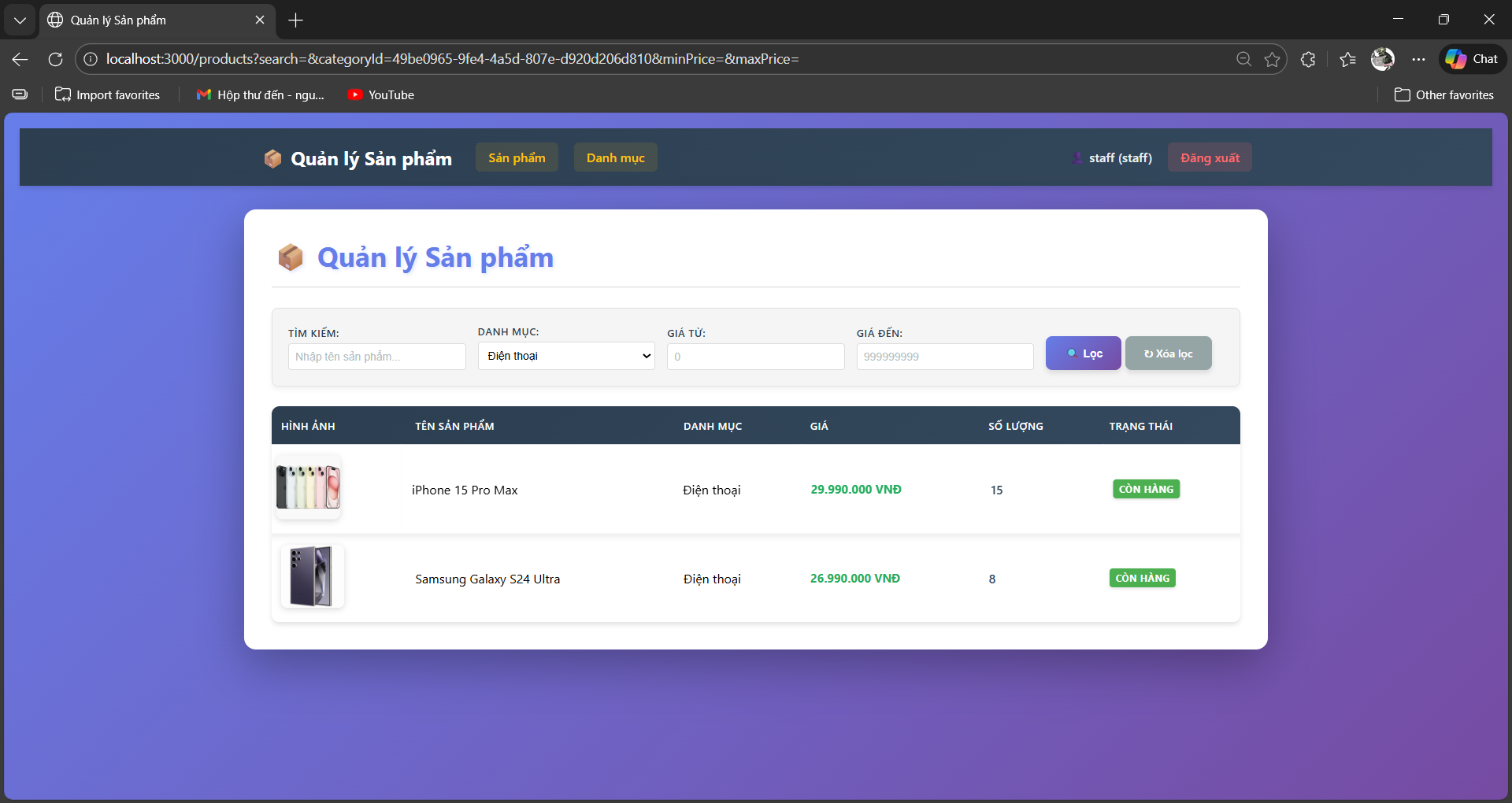


### Nhân viên

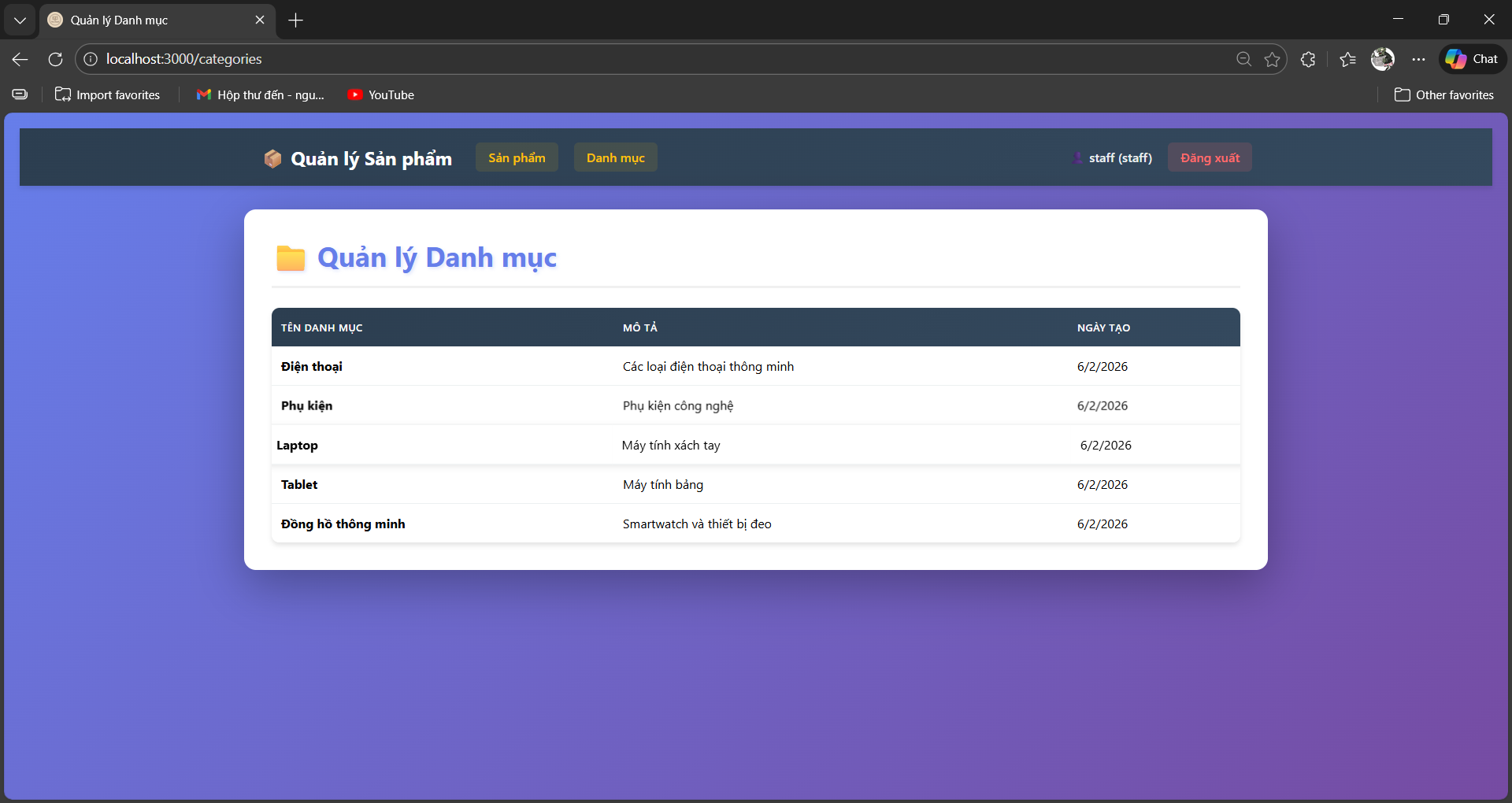
#### Màn hình sản phẩm



#### Màn hình lọc sản phẩm

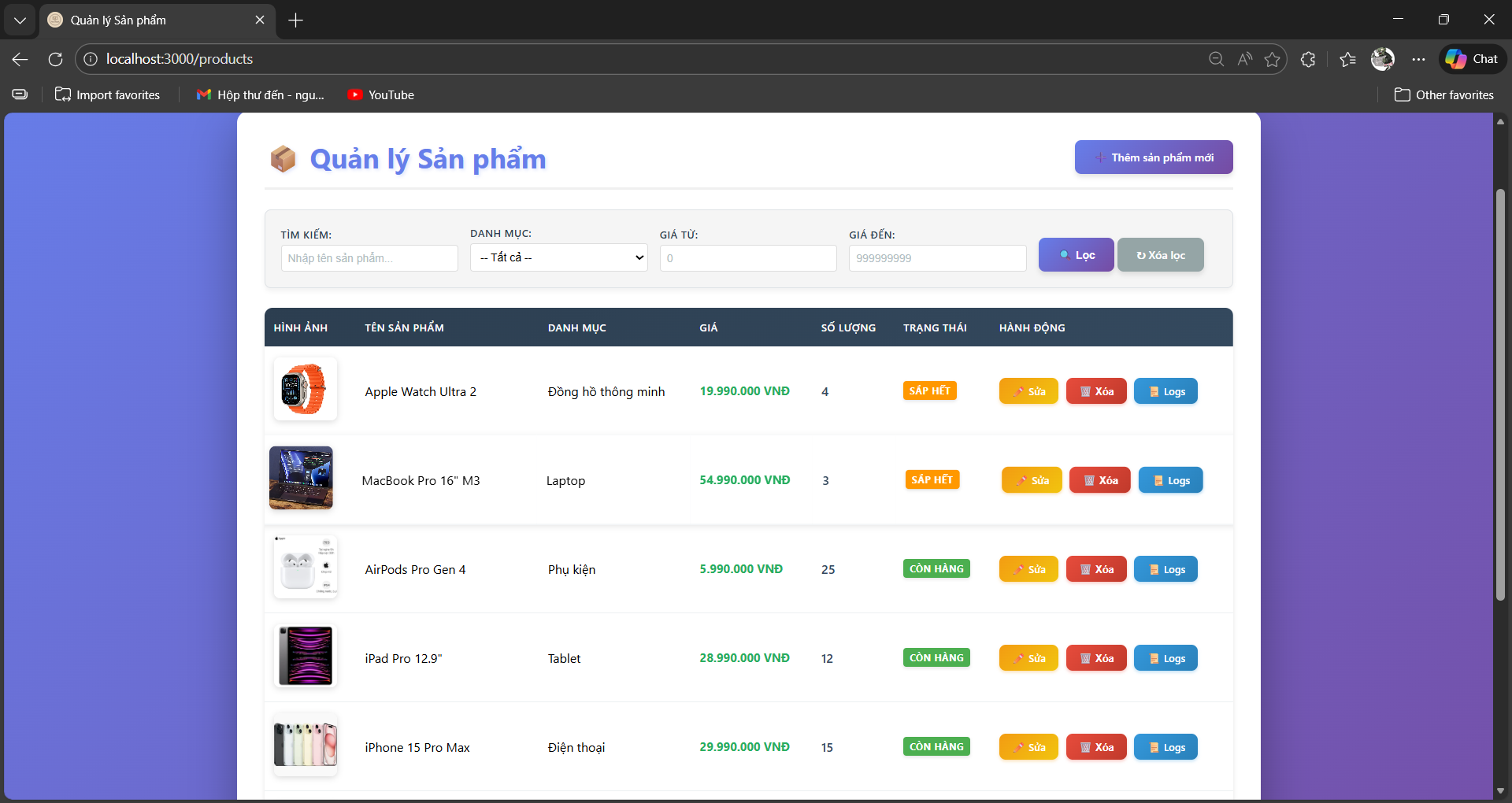


#### Màn hình danh mục

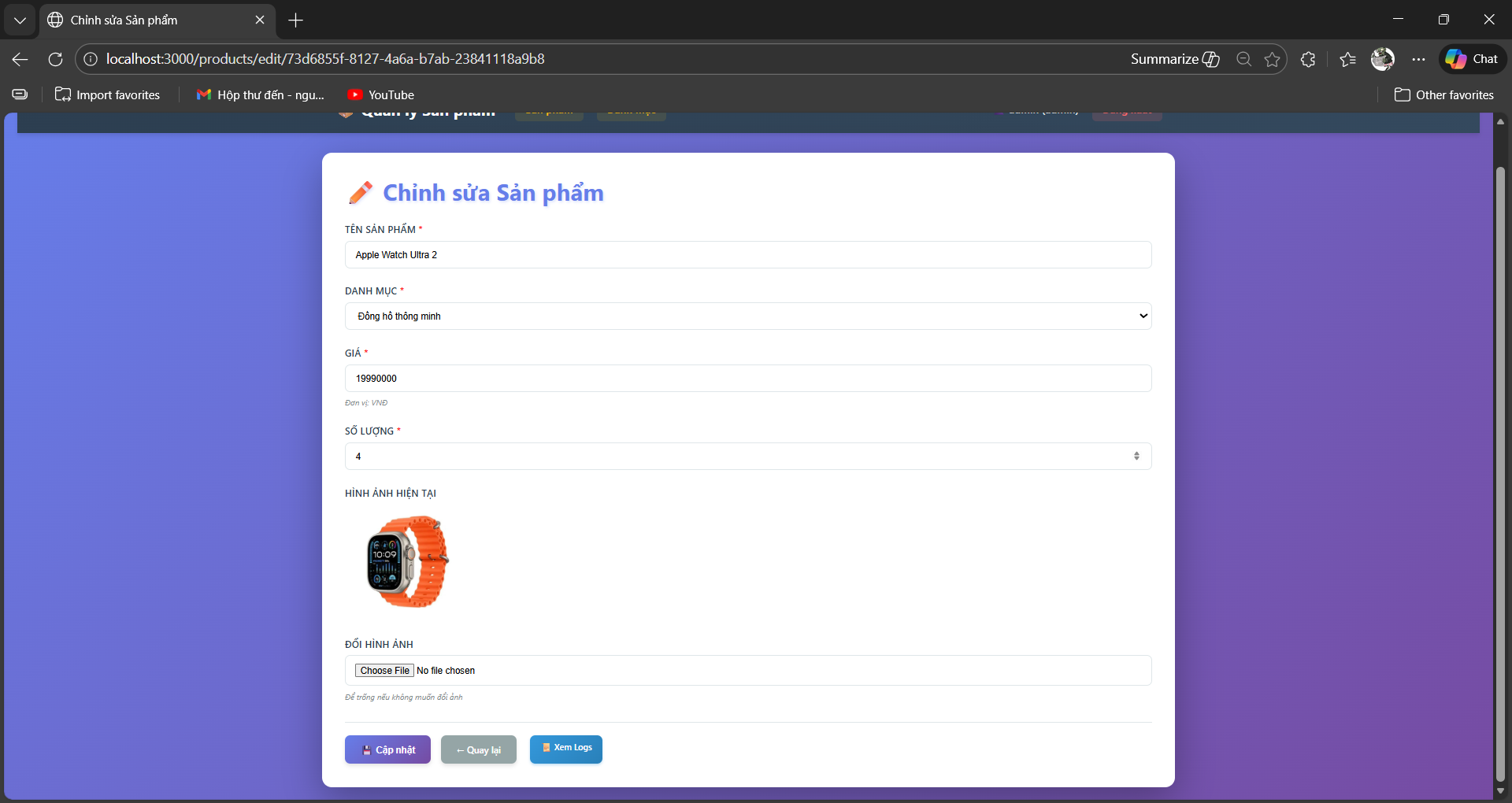


### Quản lý

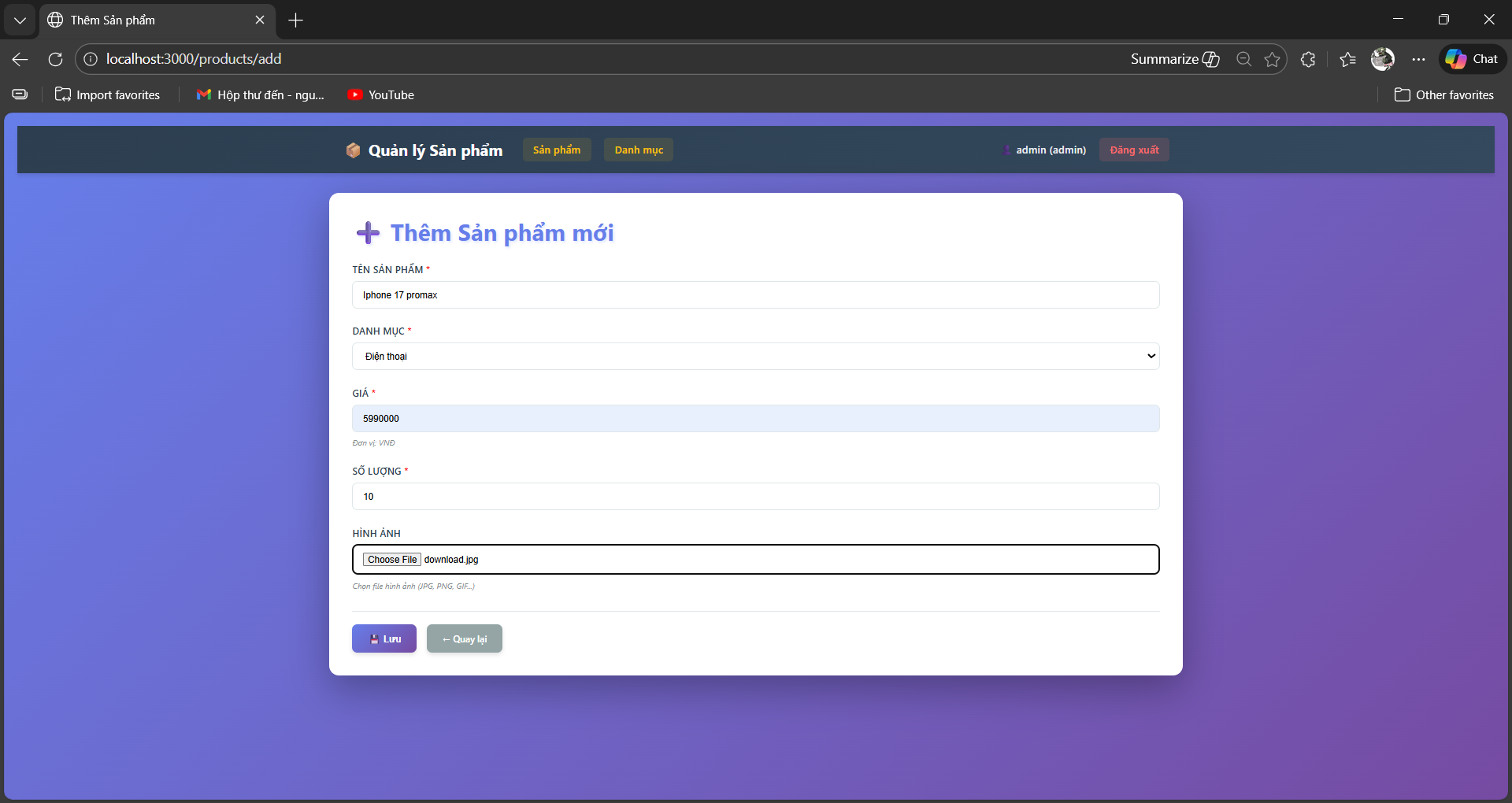
#### Màn hình quản lý sản phẩm



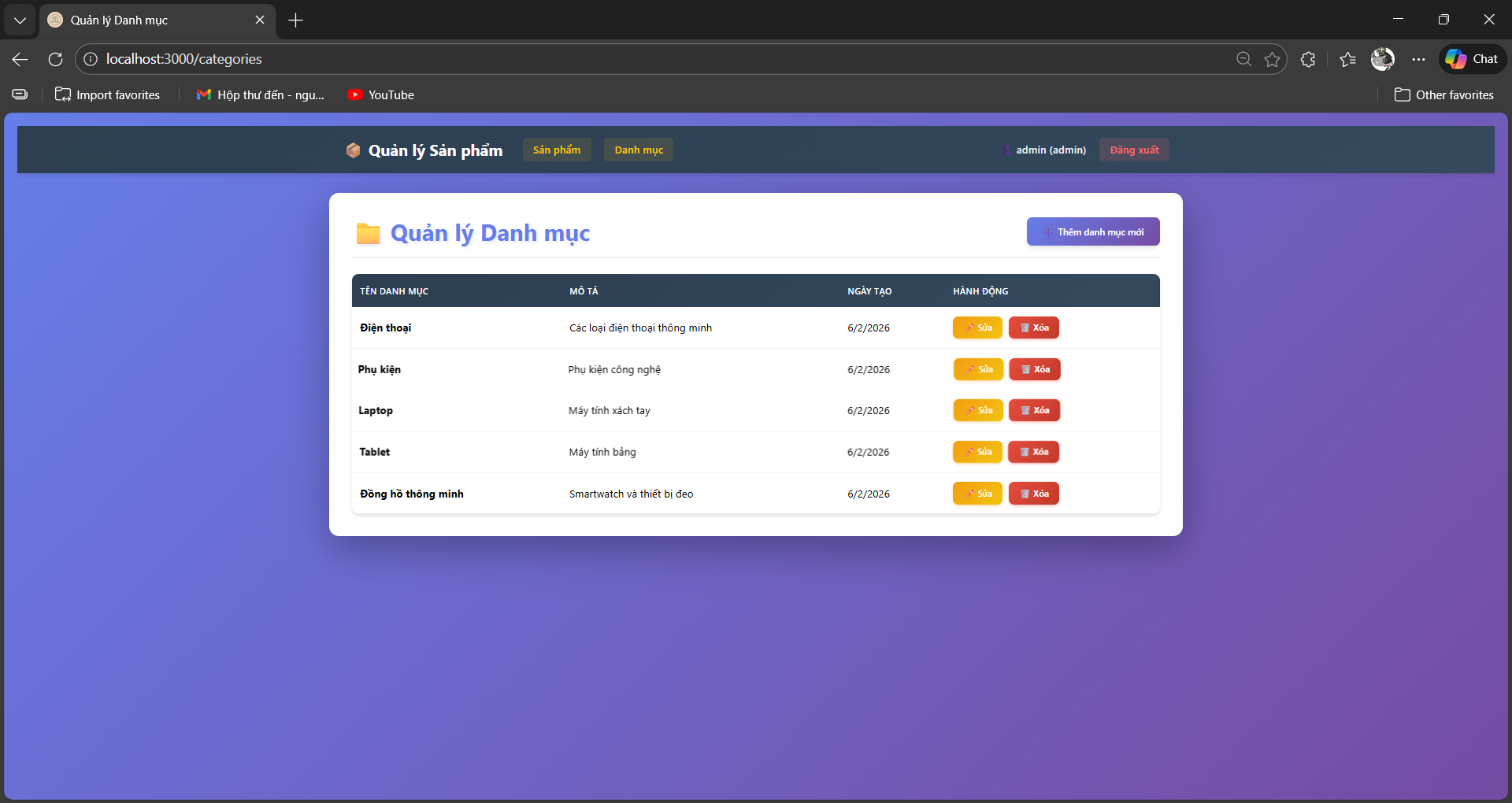
#### Màn hình chỉnh sửa sản phẩm



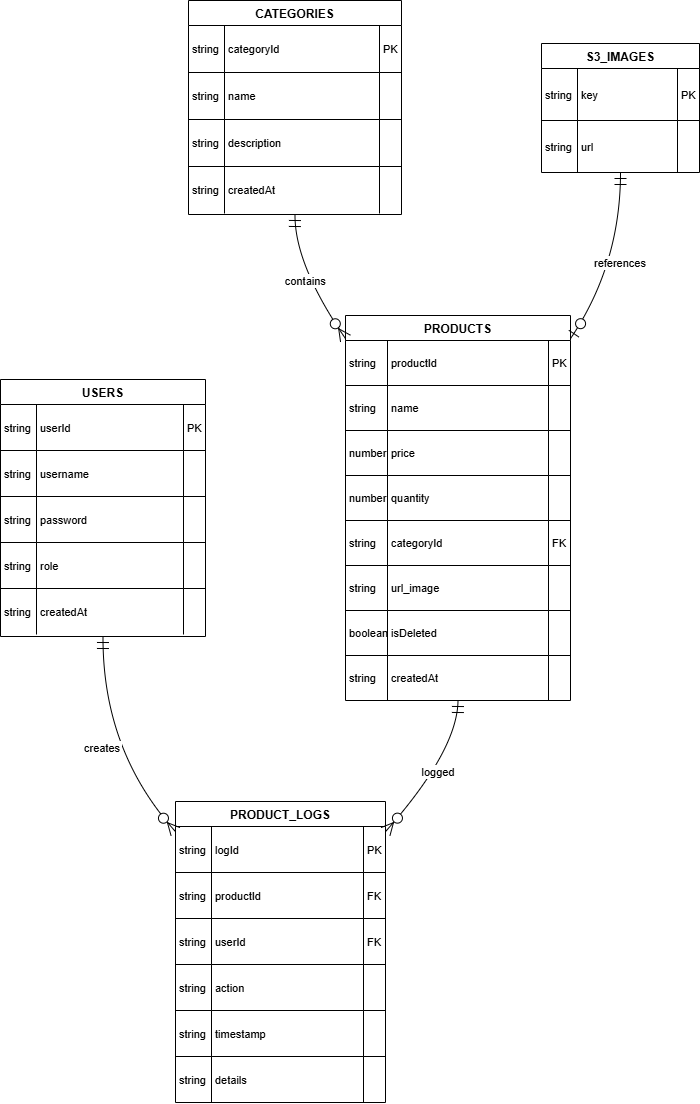
#### Màn hình thêm sản phẩm



#### Màn hình danh mục

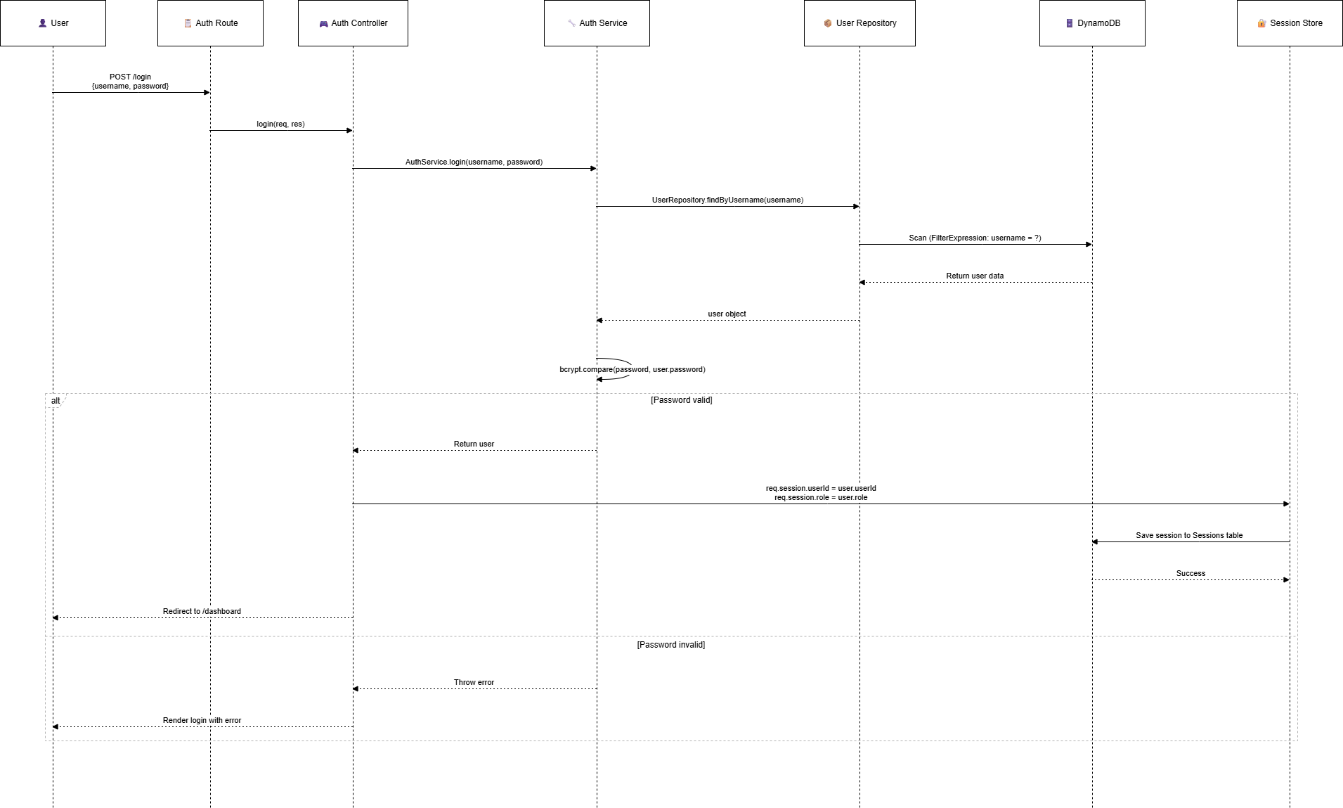


## Sơ đồ quan hệ giữa các bảng DynamoDB

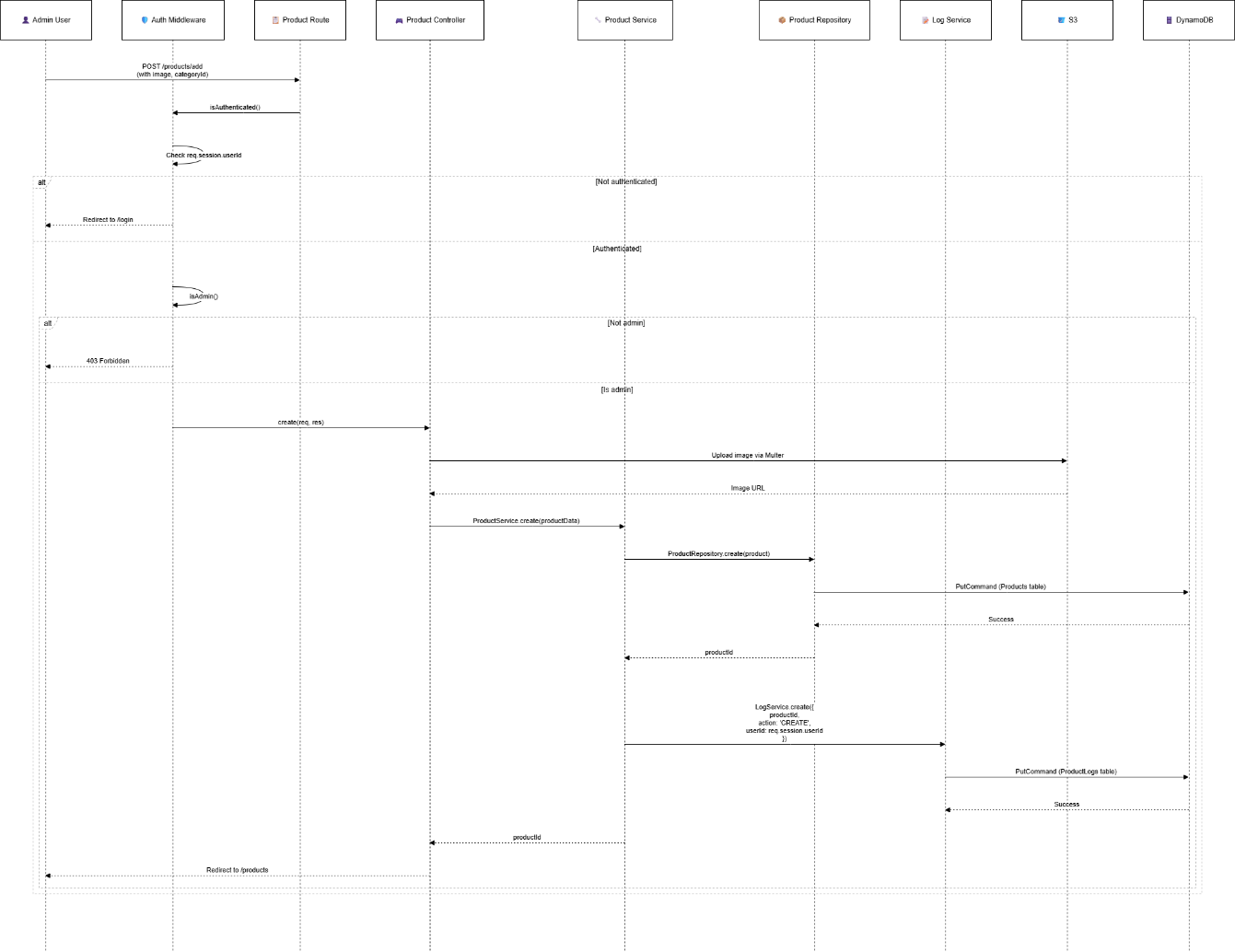


## Luồng xử lý:

### đăng nhập



### thêm sản phẩm



## So sánh:

### DynamoDB vs MySQL cho bài toán này

|  |  |  |
| --- | --- | --- |
| Tiêu chí | DynamoDB (NoSQL) | MySQL (SQL) |
| Schema | Schema-less (flexible) | Schema cứng (rigid) |
| Quan hệ | Không có JOIN, phải query nhiều lần | JOIN dễ dàng |
| Query phức tạp | Khó (Scan tốn chi phí) | Dễ (WHERE, JOIN, GROUP BY) |
| Performance | Rất nhanh cho single-item lookup (Query by PK) | Tốt với index, chậm hơn DynamoDB cho simple queries |
| Scalability | Auto-scaling, horizontal | Khó scale horizontal (cần sharding) |
| Pricing | Pay per request/capacity | Fixed cost (EC2/RDS) |
| Consistency | Eventually consistent (có thể chọn strong) | Strong consistency |
| Transaction | Hạn chế (có transaction API nhưng phức tạp) | ACID đầy đủ |
| Foreign Key | Không có | Có, enforce integrity |
| Full-text search | Không có (cần ElasticSearch) | Có (LIKE, FULLTEXT index) |

**Khi nào nên dùng DynamoDB:**

* Cần scalability cao, traffic không đều
* Data access pattern đơn giản (query by ID)
* Cần low latency (<10ms)
* Schema thay đổi thường xuyên

**Khi nào nên dùng MySQL:**

* Cần JOIN phức tạp giữa nhiều bảng
* Cần transaction đầy đủ (ACID)
* Cần aggregate functions (SUM, AVG, COUNT với GROUP BY)
* Cần full-text search nâng cao

**Cho bài toán quản lý sản phẩm này:**

* **DynamoDB phù hợp nếu:** Traffic cao, cần scale, ít query phức tạp
* **MySQL phù hợp hơn nếu:** Cần báo cáo phức tạp, nhiều JOIN (sản phẩm + category + đơn hàng + khách hàng)

## **Nhận xét ưu / nhược điểm** mô hình NoSQL

### ƯU ĐIỂM:

1. **Scalability tuyệt vời:**
   * Auto-scaling
   * Horizontal scaling (thêm partition)
   * Xử lý được hàng triệu request/second
2. **Performance cao:**
   * Latency <10ms cho Query by PK
   * Không cần index tuning phức tạp
3. **Flexible schema:**
   * Không cần migration khi thêm field
   * Mỗi item có thể có attributes khác nhau
4. **Fully managed:**
   * Không cần quản lý server
   * Auto backup, point-in-time recovery
5. **Cost-effective cho workload không đều:**
   * Pay per request (on-demand mode)
   * Không tốn tiền khi không dùng

### NHƯỢC ĐIỂM:

1. **Không có JOIN:**
   * Phải query nhiều lần
   * Code phức tạp hơn
   * Example: Hiển thị sản phẩm + category name:

// Query 1: Lấy products

const products = await ProductRepo.getAll();

// Query 2: Lấy từng category (N+1 problem)

for (let product of products) {

product.category = await CategoryRepo.getById(product.categoryId);

}

1. **Query phức tạp khó:**
   * Không có WHERE price BETWEEN x AND y AND category = 'phone' trực tiếp
   * Phải dùng Scan (tốn chi phí) hoặc thiết kế GSI phức tạp
2. **Scan operation tốn kém:**
   * Đọc toàn bộ bảng → Tốn RCU (Read Capacity Unit)
   * Chi phí tăng theo kích thước bảng
   * Example: Tìm kiếm sản phẩm theo tên = Scan toàn bộ bảng
3. **Không có aggregate functions:**
   * Không có SUM(price), AVG(quantity), COUNT(\*) GROUP BY category
   * Phải tính toán ở application layer
4. **Transaction hạn chế:**
   * TransactWriteItems chỉ support tối đa 25 items
   * Không có nested transactions
5. **Data modeling phức tạp:**
   * Phải thiết kế partition key/sort key cẩn thận
   * Không linh hoạt như SQL khi access pattern thay đổi
6. **No foreign key constraints:**
   * Application phải tự enforce integrity
   * Có thể có orphan records (sản phẩm có categoryId không tồn tại)