**Proposal for Labs and Projects**

## Contents



### Ứng Dụng Mua Sắm Trực Tuyến: Kịch Bản và Yêu Cầu

**Kịch Bản: Nền Tảng Thương Mại Điện Tử "ShopSphere"**

**ShopSphere** là một nền tảng bán lẻ trực tuyến mới, giàu tính năng, được thiết kế để cung cấp trải nghiệm mua sắm liền mạch cho khách hàng và công cụ quản lý hiệu quả cho quản trị viên. Nền tảng phải có **khả năng sẵn sàng cao, khả năng mở rộng** và **bảo mật**, đặc biệt đối với các giao dịch tài chính và dữ liệu người dùng. Nó cần hỗ trợ số lượng lớn và đang tăng của người dùng, sản phẩm và giao dịch đồng thời.

**Yêu Cầu Chính**

| **Loại** | **Mô Tả Yêu Cầu** | **Động Lực Kiến Trúc (Architectural Driver)** |
|---|---|---|
| **Chức năng** | **Quản lý Người dùng:** Đăng ký, Đăng nhập, Quản lý Hồ sơ, Lịch sử Đơn hàng. | Bảo mật, Giao diện Người dùng. |
| **Chức năng** | **Danh mục Sản phẩm:** Duyệt theo danh mục, Tìm kiếm, Xem chi tiết sản phẩm (mô tả, giá, kho hàng). | Hiệu suất, Quản lý Dữ liệu. |
| **Chức năng** | **Giỏ hàng:** Thêm, xóa, cập nhật số lượng, duy trì trạng thái giỏ hàng giữa các phiên. | Quản lý Trạng thái, Khả năng Sẵn sàng. |

| Loại | Mô Tả Yêu Cầu | Động Lực Kiến Trúc (Architectural Driver) |
|---|---|---|
| **Chức năng** | **Thanh toán & Đặt hàng:** Thanh toán nhiều bước, xử lý thanh toán an toàn (ví dụ: thẻ tín dụng, PayPal), xác nhận đơn hàng. | Bảo mật, Độ tin cậy. |
| **Chức năng** | **Xử lý Đơn hàng (Admin):** Xem, cập nhật trạng thái (Chờ xử lý, Đã giao hàng, Đã giao), hủy bỏ. | Tính toàn vẹn Giao dịch. |
| **Phi Chức năng** | **Hiệu suất:** Thời gian phản hồi cho các thao tác quan trọng (tìm kiếm sản phẩm, thanh toán) phải **dưới 2 giây**. | Khả năng mở rộng, Cân bằng Tải. |
| **Phi Chức năng** | **Bảo mật:** Mã hóa dữ liệu người dùng nhạy cảm (mật khẩu, thông tin thanh toán), xác thực an toàn (ví dụ: OAuth 2.0). | Bảo vệ Dữ liệu, Xác thực. |
| **Phi Chức năng** | **Khả năng mở rộng:** Hệ thống phải có khả năng hỗ trợ **10.000 người dùng đồng thời** và một danh mục sản phẩm đang phát triển. | Phân tán, Cô lập Thành phần. |
| **Phi Chức năng** | **Khả năng Sẵn sàng:** Yêu cầu **99.9% thời gian hoạt động** cho các tính năng mua sắm và thanh toán. | Dự phòng, Chịu lỗi. |

## Các Loại Kiến Trúc Phần mềm (Để So sánh và Sử dụng trong Lab)

Hệ thống mua sắm trực tuyến là một nghiên cứu điển hình tuyệt vời để so sánh các phương pháp kiến trúc khác nhau.

1. **Kiến Trúc Phân Tầng (Layered Architecture - Monolithic):**
    - **Mô tả:** Hệ thống được chia thành các tầng ngang: **Tầng Trình bày (UI)**, **Tầng Logic Kinh doanh**, **Tầng Truy cập Dữ liệu (Persistence)**, và **Cơ sở Dữ liệu**. Mỗi tầng chỉ tương tác với tầng ngay bên dưới nó.
    - **Trường hợp sử dụng:** Lý tưởng cho **phát triển ban đầu** hoặc các trang thương mại điện tử nhỏ hơn, nơi ưu tiên sự đơn giản trong triển khai và quản lý.
    - **Ưu điểm:** Dễ phát triển, kiểm thử và triển khai dưới dạng một đơn vị duy nhất.
2. **Kiến Trúc Microservices:**

- **Mô tả:** Ứng dụng được chia thành một tập hợp các dịch vụ nhỏ, độc lập, mỗi dịch vụ chạy trong quy trình riêng và giao tiếp thông qua các cơ chế nhẹ (như API HTTP hoặc hàng đợi tin nhắn). Các dịch vụ chính: **Dịch vụ Người dùng**, **Dịch vụ Danh mục Sản phẩm**, **Dịch vụ Giỏ hàng**, **Dịch vụ Đơn hàng**, **Dịch vụ Thanh toán**. Một **Cổng API (API Gateway)** xử lý tất cả các yêu cầu từ khách hàng.
- **Trường hợp sử dụng:** Cần thiết cho các nền tảng thương mại điện tử **quy mô lớn, sẵn sàng cao** như Amazon, cung cấp khả năng triển khai độc lập, đa dạng công nghệ và cô lập lỗi.
- **Ưu điểm:** Khả năng mở rộng cao, cô lập lỗi và thời gian ra mắt thị trường nhanh hơn cho các tính năng mới.

3. **Kiến Trúc Hướng Sự Kiện (Event-Driven Architecture - EDA):**
   - **Mô tả:** Các thành phần giao tiếp chủ yếu thông qua các sự kiện, thường sử dụng một broker tin nhắn (ví dụ: Kafka, RabbitMQ). Các thay đổi trạng thái (sự kiện) được xuất bản bởi một dịch vụ (người sản xuất) và được tiêu thụ bởi các dịch vụ khác (người tiêu thụ) quan tâm đến thay đổi đó.
   - **Trường hợp sử dụng:** Tuyệt vời để **tách biệt khối lượng công việc giao dịch và phân tích**, chẳng hạn như cập nhật kho hàng và gửi email thông báo *sau khi* đơn hàng được đặt.
   - **Ưu điểm:** Khả năng phản hồi theo thời gian thực, khả năng tách biệt cao và dễ dàng tích hợp các dịch vụ tiêu thụ mới.

---

## 8 Bài Thực Hành (Lab)

8 bài thực hành sau đây sử dụng kịch bản **ShopSphere**, tiến triển từ phân tích yêu cầu đến triển khai một phần, so sánh kiến trúc **Phân Tầng** và **Microservices**.

| Lab | Tiêu đề | Trọng tâm Kiến trúc | Hoạt động & Hướng dẫn Từng bước Chi tiết |
|---|---|---|---|
| **Lab 1** | **Thu thập Yêu cầu & Mô hình hóa (Use Case)** | N/A (Tiền Kiến trúc) | **Hoạt động:** Xác định Yêu cầu Chức năng/Phi Chức năng. Vẽ Biểu đồ Use Case UML cho các tác nhân **"Khách hàng Web"** và **"Quản trị viên"**. **Các bước:** 1. Xác định tất cả các tác nhân chính. 2. Xác định các use case cấp cao (ví dụ: Thực hiện Mua hàng, Quản lý Sản phẩm). 3. Chi tiết hóa một use case quan trọng |

| Lab | Tiêu đề | Trọng tâm Kiến trúc | Hoạt động & Hướng dẫn Từng bước Chi tiết |
|---|---|---|---|
| | | | (ví dụ: "Thanh toán") với các điều kiện trước/sau và luồng sự kiện cơ bản. 4. Xác định ba Yêu cầu Quan trọng về Kiến trúc (ASR). |
| **Lab 2** | **Thiết kế Kiến trúc Phân Tầng (Tầm nhìn Logic)** | Phân Tầng | **Hoạt động:** Thiết kế kiến trúc 4 tầng. **Các bước:** 1. Định nghĩa bốn tầng: Presentation, Business Logic, Persistence, Database. 2. Chỉ định các thành phần cốt lõi trong mỗi tầng (ví dụ: ProductController trong Presentation, OrderManager trong Business Logic, SQLProductRepo trong Persistence). 3. Vẽ biểu đồ thành phần hiển thị sự phụ thuộc của các tầng (luồng nghiêm ngặt từ trên xuống). |
| **Lab 3** | **Triển khai Kiến trúc Phân Tầng (CRUD)** | Phân Tầng | **Hoạt động:** Triển khai tính năng **Quản lý Sản phẩm** bằng mẫu Phân Tầng. **Các bước:** 1. Thiết lập cấu trúc dự án cơ bản (ví dụ: Java/Spring hoặc Python/Flask) với các gói cho mỗi tầng. 2. Triển khai thực thể Product. 3. Viết code cho **Tầng Persistence** để kết nối với cơ sở dữ liệu giả/trong bộ nhớ (các phương thức Create, Read, Update, Delete). 4. Triển khai **Tầng Business Logic** để xác thực sản phẩm. 5. Triển khai **Tầng Presentation** (Controller) để hiển thị một endpoint REST (/products). |
| **Lab 4** | **Phân rã Microservices & Giao tiếp** | Microservices | **Hoạt động:** Phân rã Monolith và xác định hợp đồng dịch vụ. **Các bước:** 1. Xác định năm dịch vụ cốt lõi (**User, Product, Cart, Order, Payment**). 2. Định nghĩa các endpoint API chính (Hợp đồng Dịch vụ) cho **Dịch vụ Sản phẩm** và **Dịch vụ Giỏ hàng**. 3. Vẽ Mô hình C4 (Cấp độ 1: Bối cảnh Hệ thống) hiển thị hệ thống và các phụ thuộc bên ngoài (Cổng Thanh toán, Dịch vụ Email). |

| Lab | Tiêu đề | Trọng tâm Kiến trúc | Hoạt động & Hướng dẫn Từng bước Chi tiết |
|---|---|---|---|
| **Lab 5** | **Triển khai Dịch vụ Microservice Sản phẩm** | Microservices | **Hoạt động:** Xây dựng một Dịch vụ Microservice Sản phẩm độc lập. **Các bước:** 1. Tạo một dự án dịch vụ độc lập mới (cổng/kho lưu trữ khác). 2. Triển khai **Dịch vụ Sản phẩm** (Logic quản lý Danh mục và Kho hàng). 3. Hiển thị API REST để đọc và tìm kiếm sản phẩm (GET /api/products). 4. Kiểm tra dịch vụ một cách độc lập bằng một công cụ như Postman. |
| **Lab 6** | **Giới thiệu Mẫu Cổng API (API Gateway)** | Microservices | **Hoạt động:** Triển khai một Cổng API đơn giản. **Các bước:** 1. Sử dụng một framework (ví dụ: Spring Cloud Gateway, Nginx) hoặc viết một proxy ngược cơ bản. 2. Cấu hình gateway để định tuyến các yêu cầu /api/products tới **Dịch vụ Sản phẩm** (từ Lab 5). 3. Triển khai một kiểm tra bảo mật duy nhất (ví dụ: xác thực token stub) trên gateway trước khi chuyển tiếp yêu cầu. |
| **Lab 7** | **Kiến trúc Hướng Sự kiện (EDA) & Tích hợp** | Hướng Sự kiện | **Hoạt động:** Triển khai giao tiếp không đồng bộ bằng sự kiện. **Các bước:** 1. Thiết lập một broker tin nhắn cục bộ (ví dụ: RabbitMQ hoặc thiết lập Kafka giả). 2. Sửa đổi **Dịch vụ Đơn hàng** để **xuất bản** một OrderPlacedEvent khi thanh toán thành công. 3. Tạo một **Dịch vụ Thông báo** mới **đăng ký** OrderPlacedEvent và in một thông báo xác nhận (mô phỏng email). 4. Chứng minh bản chất tách rời của các dịch vụ. |
| **Lab 8** | **Tầm nhìn Triển khai & Phân tích Thuộc tính Chất lượng (ATAM)** | Hỗn hợp/Triển khai | **Hoạt động:** Thiết kế triển khai hệ thống và đánh giá Yêu cầu Phi Chức năng. **Các bước:** 1. Vẽ Biểu đồ Triển khai UML hiển thị các Microservices trên container/VM, Cổng API và bộ cân bằng tải. 2. Chọn ba Yêu cầu Phi Chức năng (ví dụ: Khả năng mở rộng, Bảo mật, Hiệu suất). 3. Tiến hành Phân tích Đánh đổi Kiến trúc (ATAM) đơn giản để thảo luận cách các kiến trúc **Microservices** và **Phân Tầng** xử lý một yêu cầu |

| Lab | Tiêu đề | Trọng tâm Kiến trúc | Hoạt động & Hướng dẫn Từng bước Chi tiết |
|---|---|---|---|
| | | | đã chọn (ví dụ: Khả năng mở rộng) khác nhau như thế nào. |

---

## Các Dự Án Khác

Dưới đây là bốn kịch bản được xác định đầy đủ khác cho các dự án khóa học, từ độ phức tạp sơ cấp đến trung bình, bao gồm các bước hướng dẫn chi tiết và bước lập trình.

### Dự án 1: Ứng Dụng Danh Sách Việc Cần Làm (To-Do List) (Cơ bản)

- **Kịch bản:** Một công cụ năng suất cá nhân đơn giản để quản lý các nhiệm vụ hàng ngày. Người dùng có thể thêm, xem, đánh dấu là hoàn thành và xóa nhiệm vụ.
- **Trọng tâm Kiến trúc: Kiến trúc Máy khách-Máy chủ** (1-Tầng, tất cả trên một máy/trình duyệt ban đầu).
- **Các bước Hướng dẫn:**
    1. **Xác định Mô hình:** Tạo một lớp (hoặc cấu trúc) Task với id, description, và is_completed.
    2. **Phát triển Giao diện:** Thiết kế một biểu mẫu web hoặc giao diện điều khiển đơn giản để nhập.
    3. **Triển khai CRUD:** Viết các hàm để **T**ạo (Create), **Đ**ọc (Read - hiển thị tất cả), **C**ập nhật (Update - đánh dấu hoàn thành), và **X**óa (Delete) các nhiệm vụ khỏi một danh sách trong bộ nhớ (hoặc một tệp JSON đơn giản).
- **Các bước Lập trình (Ví dụ Python/Flask):**
    - **Cấu trúc Dữ liệu:** Sử dụng một list Python để lưu trữ các từ điển đại diện cho nhiệm vụ: tasks = [{'id': 1, 'desc': 'Đi chợ', 'complete': False}].
    - **Hàm Thêm:** Định nghĩa một hàm add_task(description) tạo ID mới và nối nhiệm vụ mới vào danh sách.
    - **Hiển thị:** Định nghĩa một hàm lặp qua danh sách và in/render từng nhiệm vụ.

## Dự án 2: Hệ Thống Quản lý Điểm Sinh viên Đơn giản (Cơ bản-Trung bình)

- **Kịch bản:** Một hệ thống cho một phòng ban nhỏ để quản lý việc đăng ký sinh viên, hồ sơ khóa học và giao/xem điểm. Người dùng Quản trị có thể thêm sinh viên và khóa học, và giảng viên có thể nhập điểm.
- **Trọng tâm Kiến trúc: Kiến trúc Phân Tầng (2-Tầng)**: **Máy khách (Web UI)** và **Máy chủ (Logic Kinh doanh + Persistence trong tệp/Cơ sở dữ liệu)**.
- **Các bước Hướng dẫn:**
    1. **Xác định Mô hình:** Tạo Student, Course, và Enrollment (liên kết sinh viên, khóa học và lưu trữ grade).
    2. **Thiết kế Cơ sở Dữ liệu:** Phác thảo một lược đồ cơ sở dữ liệu quan hệ đơn giản (3 bảng) và khóa chính/khóa ngoại.
    3. **Triển khai Persistence:** Sử dụng một tệp (CSV/JSON) hoặc một cơ sở dữ liệu SQLite đơn giản để lưu trữ dữ liệu.
    4. **Triển khai Logic Kinh doanh:** Phát triển các hàm để tính điểm trung bình (GPA) của sinh viên dựa trên các khóa học đã đăng ký của họ.
- **Các bước Lập trình (Ví dụ SQL/Python):**
    - **Lược đồ SQL:** CREATE TABLE Students (id INT PRIMARY KEY, name TEXT); CREATE TABLE Courses (code TEXT PRIMARY KEY, name TEXT); CREATE TABLE Enrollments (student_id INT, course_code TEXT, grade REAL, FOREIGN KEY...);
    - **Hàm GPA:** Viết một hàm calculate_gpa(student_id) thực thi một truy vấn SQL (hoặc logic đọc tệp) để lấy tất cả điểm của một sinh viên và tính điểm trung bình có trọng số.

## Dự án 3: Dịch Vụ Rút Gọn URL (Trung bình)

- **Kịch bản:** Một dịch vụ giống như Bitly hoặc tinyurl.com lấy một URL dài và tạo ra một mã ngắn, duy nhất (ví dụ: bit.ly/AbCde12). Khi truy cập liên kết ngắn, người dùng được chuyển hướng đến URL dài ban đầu. Phải theo dõi số lần nhấp.
- **Trọng tâm Kiến trúc: Kiến trúc Máy khách-Máy chủ (3-Tầng/N-Tầng)**: **Máy khách**, **Máy chủ Web/Logic Ứng dụng**, **Cơ sở Dữ liệu**. Tập trung vào hiệu suất đọc cao.
- **Các bước Hướng dẫn:**
    1. **Tính năng Cốt lõi:** Triển khai logic rút gọn URL (tạo một khóa chữ và số ngắn, duy nhất).

2. **Chuyển hướng:** Cấu hình máy chủ web/ứng dụng để xử lý mã ngắn như một tuyến đường và tra cứu URL dài.
3. **Phân tích:** Thêm một cột vào cơ sở dữ liệu để theo dõi số lần liên kết ngắn đã được nhấp.
4. **Bảo mật/Đầu vào:** Triển khai xác thực URL cơ bản để ngăn chặn các URL không phải HTTP/HTTPS.

- **Các bước Lập trình (Ví dụ về bất kỳ Framework Backend nào):**
    - **Tạo Khóa:** Sử dụng một hàm như base62_encode hoặc một hàm băm đơn giản để tạo mã ngắn duy nhất từ một ID tăng dần.
    - **Cơ sở Dữ liệu:** CREATE TABLE urls (id INT PRIMARY KEY, short_code VARCHAR(10) UNIQUE, long_url TEXT, click_count INT DEFAULT 0);
    - **Tuyến đường Chuyển hướng:** Định nghĩa một tuyến đường (ví dụ: /u/<short_code>) tra cứu long_url, tăng click_count, và trả về phản hồi **301/302 HTTP Redirect**.

### Dự án 4: Ứng Dụng Chat Thời Gian Thực (Trung bình)

- **Kịch bản:** Một ứng dụng chat đa người dùng đơn giản nơi người dùng có thể tham gia một phòng và gửi tin nhắn đến tất cả những người dùng khác trong phòng đó. Tin nhắn phải xuất hiện ngay lập tức mà không cần làm mới trang.
- **Trọng tâm Kiến trúc: Kiến trúc Hướng Sự kiện (sử dụng WebSockets)**. Tin nhắn là các sự kiện được đẩy từ máy chủ đến tất cả các máy khách đã kết nối.
- **Các bước Hướng dẫn:**
    1. **Thiết lập Máy chủ WebSocket:** Cấu hình một backend để hỗ trợ giao thức WebSocket (ví dụ: Socket.io, Flask-SocketIO hoặc WebSockets thô).
    2. **Kết nối Máy khách:** Triển khai một máy khách HTML/JavaScript cơ bản để mở kết nối WebSocket.
    3. **Truyền tin nhắn:** Máy chủ phải nhận một tin nhắn từ một máy khách và **truyền tin (broadcast)** nó đến tất cả các máy khách đang hoạt động khác trong phòng.
    4. **Persistence (Tùy chọn):** Thêm tính năng lưu trữ để lưu 10 tin nhắn cuối cùng vào cơ sở dữ liệu.
- **Các bước Lập trình (Ví dụ Node.js/Socket.io):**

- **Logic Máy chủ:** Khi nhận được sự kiện message: io.on('connection', (socket) => { socket.on('chat message', (msg) => { io.emit('chat message', msg); }); }); (Lệnh io.emit truyền tin đến mọi người).
- **Logic Máy khách (JS):** Sử dụng const socket = io(); để kết nối và socket.on('chat message', function(msg) { // thêm tin nhắn vào màn hình }); để nhận và hiển thị.