**BÁO CÁO DỰ ÁN PRJ301**

**Tên dự án: GSPMK**

**VŨ ĐÌNH ĐĂNG – SE1888**

# Phần I: Cơ sở lý thuyết

## 1. Tổng quan về công nghệ Java Servlet

Java Servlet là một công nghệ được phát triển bởi Sun Microsystems (nay là Oracle), đóng vai trò như một lớp trung gian giữa ứng dụng web và máy chủ web. Java Servlet thường được triển khai trong các ứng dụng Java Enterprise Edition (Jakarta EE) để xử lý các yêu cầu HTTP từ trình duyệt và trả về phản hồi tương ứng. Trong bối cảnh lập trình web, Servlet hoạt động ở cấp độ máy chủ, tương tác với các yêu cầu của khách hàng, xử lý dữ liệu từ yêu cầu đó và gửi lại kết quả.

**Các thành phần cơ bản của Java Servlet:**

* **HttpServlet**: Lớp cơ bản của tất cả các Servlet trong ứng dụng Java. HttpServlet cung cấp các phương thức như doGet(), doPost(), doPut(), doDelete(), v.v., để xử lý các loại yêu cầu HTTP tương ứng.
* **Servlet Container**: Là môi trường mà Servlet chạy trong đó (ví dụ: Apache Tomcat, GlassFish, WildFly). Container này chịu trách nhiệm quản lý vòng đời của Servlet, từ khi nó được khởi tạo đến khi nó bị hủy.
* **Servlet Config và Servlet Context**: Hai đối tượng này giúp Servlet lưu trữ và truy xuất thông tin cấu hình (Servlet Config) và thông tin chung của ứng dụng (Servlet Context).

**Vòng đời của Java Servlet:**

Vòng đời của một Servlet bao gồm ba giai đoạn chính: **Khởi tạo (Initialization)**, **Xử lý yêu cầu (Request Handling)** và **Hủy (Destruction)**. Khi yêu cầu đầu tiên đến, container tạo ra một thể hiện của Servlet và gọi phương thức init(). Sau đó, với mỗi yêu cầu, phương thức service() sẽ được gọi và chuyển tiếp tới phương thức tương ứng (doGet(), doPost()). Khi không còn cần thiết, container sẽ gọi destroy() để giải phóng tài nguyên.

## 2. AJAX và XMLHTTPRequest

**AJAX (Asynchronous JavaScript and XML)** là một kỹ thuật cho phép ứng dụng web gửi và nhận dữ liệu từ máy chủ mà không cần tải lại toàn bộ trang web. AJAX sử dụng đối tượng XMLHttpRequest để thực hiện các yêu cầu HTTP bất đồng bộ. Trong bối cảnh Java Servlet, AJAX thường được dùng để gọi các Servlet xử lý một phần dữ liệu (ví dụ: tải dữ liệu động, kiểm tra đầu vào) mà không cần tải lại trang.

**Các bước thực hiện AJAX với Java Servlet:**

1. **Khởi tạo XMLHttpRequest**: Ở phía JavaScript, tạo một thể hiện của đối tượng XMLHttpRequest.
2. **Mở và gửi yêu cầu**: Sử dụng phương thức open() để cấu hình yêu cầu và send() để gửi yêu cầu đến Servlet.
3. **Xử lý phản hồi**: Khi phản hồi từ Servlet được trả về, phương thức onreadystatechange sẽ được gọi, cho phép xử lý dữ liệu phản hồi.

**Ứng dụng thực tiễn của AJAX trong Servlet:**

* **Tải nội dung động**: Ví dụ, trong một ứng dụng bán hàng, AJAX có thể được sử dụng để cập nhật giỏ hàng mà không cần tải lại trang.
* **Kiểm tra đầu vào ngay lập tức**: AJAX có thể được dùng để xác minh tên người dùng hoặc mật khẩu trước khi gửi biểu mẫu.
* **Hiển thị dữ liệu thời gian thực**: AJAX cho phép làm mới các phần của trang web theo thời gian thực, như cập nhật số lượng người xem.

## 3. WebSocket, SSE và AJAX Polling

Trong lập trình web, các kỹ thuật **WebSocket**, **Server-Sent Events (SSE)**, và **AJAX Polling** thường được sử dụng để truyền tải dữ liệu theo thời gian thực giữa máy khách và máy chủ. Trong Java Servlet, các kỹ thuật này giúp tăng tính tương tác cho ứng dụng web.

**WebSocket:**

WebSocket là một giao thức giúp tạo kết nối hai chiều giữa máy khách và máy chủ, hỗ trợ trao đổi dữ liệu thời gian thực với độ trễ thấp. So với các yêu cầu HTTP thông thường, WebSocket duy trì kết nối mở, cho phép trao đổi dữ liệu không đồng bộ mà không cần khởi tạo lại kết nối. Trong Jakarta EE, WebSocket API cho phép tích hợp WebSocket vào ứng dụng Servlet dễ dàng.

**SSE (Server-Sent Events):**

SSE là một công nghệ truyền tải dữ liệu từ máy chủ tới máy khách theo thời gian thực. SSE phù hợp để gửi dữ liệu từ máy chủ đến máy khách trong khi máy khách không cần gửi lại yêu cầu liên tục. Đối với Java Servlet, các Servlet có thể sử dụng các OutputStream để truyền dữ liệu SSE liên tục đến máy khách.

**AJAX Polling:**

AJAX Polling là kỹ thuật mà máy khách thường xuyên gửi yêu cầu đến máy chủ để kiểm tra xem có dữ liệu mới không. Dù không hiệu quả bằng WebSocket hoặc SSE (do tốn tài nguyên khi phải gửi nhiều yêu cầu), AJAX Polling lại dễ triển khai.

**So sánh các kỹ thuật trong Java Servlet:**

* **WebSocket**: Hiệu quả và linh hoạt nhất cho dữ liệu hai chiều.
* **SSE**: Dễ triển khai cho dữ liệu một chiều (server-to-client).
* **AJAX Polling**: Đơn giản nhưng tốn nhiều tài nguyên nếu số lượng yêu cầu lớn.

## 4. Listener và Session tự tạo

**Listener trong Java Servlet:**

Listener là các đối tượng đặc biệt trong Java Servlet, dùng để giám sát và phản hồi các sự kiện xảy ra trong ứng dụng. Listener cho phép xử lý các sự kiện khi có sự thay đổi trong vòng đời của ứng dụng web, phiên làm việc (session), và yêu cầu.

* **ServletContextListener**: Theo dõi sự kiện khi ứng dụng được khởi động hoặc dừng.
* **HttpSessionListener**: Theo dõi khi phiên làm việc mới được tạo hoặc kết thúc.
* **ServletRequestListener**: Theo dõi khi yêu cầu mới được tạo hoặc bị hủy.

**Session tự tạo:**

Session (phiên làm việc) là một cơ chế trong Java Servlet giúp duy trì thông tin của người dùng giữa các yêu cầu HTTP. Các servlet có thể tạo và quản lý session bằng cách sử dụng HttpSession và gán thông tin người dùng vào session này. Việc tạo session tự tạo yêu cầu cài đặt các thuộc tính của session sao cho phù hợp với nhu cầu của ứng dụng.

**Quản lý session trong ứng dụng Java Servlet:**

* **Tạo session**: Sử dụng request.getSession() để tạo hoặc lấy session hiện tại.
* **Lưu dữ liệu trong session**: Sử dụng session.setAttribute() để lưu trữ dữ liệu.
* **Hủy session**: Sử dụng session.invalidate() để hủy session khi không còn cần thiết.

**Ví dụ thực tiễn của Listener và Session trong Java Servlet:**

Listener và session thường được sử dụng trong các ứng dụng yêu cầu tính bảo mật cao, như hệ thống quản lý người dùng hoặc thương mại điện tử. Session giúp duy trì trạng thái của người dùng, trong khi Listener giúp theo dõi các sự kiện, chẳng hạn như đăng nhập và đăng xuất, để thực hiện các hành động như ghi nhật ký hoặc cập nhật dữ liệu người dùng.

## 5, Tổng quan về Hibernate

Hibernate được sử dụng rộng rãi trong các ứng dụng Java EE nhờ vào khả năng giảm thiểu mã SQL thủ công và tăng tính hiệu quả trong quản lý dữ liệu. Hibernate tự động chuyển đổi các đối tượng Java thành dữ liệu có thể lưu trữ trong các bảng cơ sở dữ liệu và ngược lại.

Hibernate có thể hoạt động với hầu hết các hệ quản trị cơ sở dữ liệu phổ biến như MySQL, PostgreSQL, Oracle, và SQL Server, và thậm chí còn hỗ trợ các hệ quản trị phi quan hệ như MongoDB.

**Những điểm nổi bật của Hibernate:**

* **Tính di động**: Vì Hibernate độc lập với nền tảng, nên ứng dụng có thể chuyển đổi dễ dàng giữa các loại cơ sở dữ liệu mà không cần thay đổi nhiều mã nguồn.
* **Quản lý vòng đời của đối tượng**: Hibernate giúp quản lý trạng thái của đối tượng trong suốt vòng đời của nó, từ lúc được tạo ra đến khi bị xóa.
* **Lazy Loading và Eager Loading**: Hibernate cung cấp cơ chế tải dữ liệu linh hoạt, giúp quản lý tài nguyên tốt hơn bằng cách chỉ tải dữ liệu khi cần thiết (Lazy Loading) hoặc tải toàn bộ dữ liệu ngay lập tức (Eager Loading).

**Cấu trúc và Cách hoạt động của Hibernate**

**1. Hibernate Configuration File (hibernate.cfg.xml):** Đây là file cấu hình chính của Hibernate, nơi lưu trữ các thông tin cấu hình về cơ sở dữ liệu (URL, tên người dùng, mật khẩu), cũng như các thuộc tính tùy chỉnh khác của Hibernate.

**2. Mapping File (hbm.xml):** Hibernate hỗ trợ ánh xạ giữa lớp Java và bảng cơ sở dữ liệu thông qua các file .hbm.xml, hoặc bằng cách sử dụng các annotation. Các file này định nghĩa cách Hibernate sẽ ánh xạ các thuộc tính của lớp Java sang các cột của bảng cơ sở dữ liệu.

**3. Session và SessionFactory:** Session là đối tượng làm việc chính của Hibernate, chịu trách nhiệm quản lý các thao tác CRUD (Create, Read, Update, Delete). SessionFactory là đối tượng chính để khởi tạo và cấu hình các phiên làm việc, quản lý các phiên kết nối với cơ sở dữ liệu. SessionFactory thường được tạo một lần khi ứng dụng khởi chạy và cung cấp các Session theo yêu cầu.

**Các Chức năng Chính của Hibernate**

**1. CRUD (Create, Read, Update, Delete):** Hibernate cung cấp các phương thức trực tiếp để tạo, đọc, cập nhật và xóa các đối tượng trong cơ sở dữ liệu mà không cần viết câu lệnh SQL thủ công.

**2. HQL (Hibernate Query Language):** HQL là ngôn ngữ truy vấn của Hibernate, gần giống với SQL nhưng sử dụng các tên lớp và thuộc tính Java thay vì các tên bảng và cột. Điều này làm cho HQL có tính di động cao hơn vì nó không phụ thuộc vào hệ quản trị cơ sở dữ liệu cụ thể.

**3. Caching (Bộ nhớ đệm):** Hibernate hỗ trợ cơ chế bộ nhớ đệm (Caching) để tăng hiệu suất bằng cách lưu trữ các bản ghi đã truy xuất. Hibernate có hai cấp độ bộ nhớ đệm:

* **First-level Cache**: Bộ nhớ đệm cấp đầu tiên, được bật mặc định, áp dụng cho từng phiên làm việc (Session).
* **Second-level Cache**: Bộ nhớ đệm cấp thứ hai, cho phép chia sẻ dữ liệu giữa các phiên làm việc khác nhau. Cần cài đặt thêm các thư viện như Ehcache, Infinispan để sử dụng chức năng này.

**4. Transaction Management (Quản lý giao dịch):** Hibernate hỗ trợ quản lý giao dịch, đảm bảo rằng các thao tác với cơ sở dữ liệu là nguyên tử, tức là nếu một phần của giao dịch thất bại, toàn bộ giao dịch sẽ bị hủy bỏ để giữ tính toàn vẹn của dữ liệu.

**Các Annotation trong Hibernate**

Ngoài cách ánh xạ với file XML, Hibernate cũng hỗ trợ ánh xạ với annotation, đơn giản hóa quá trình ánh xạ và làm cho mã dễ đọc hơn. Một số annotation phổ biến:

* @Entity: Đánh dấu một lớp là thực thể Hibernate (một bảng trong cơ sở dữ liệu).
* @Table: Xác định tên bảng mà thực thể ánh xạ tới.
* @Id: Chỉ định thuộc tính nào là khóa chính.
* @GeneratedValue: Đặt chế độ tự động tạo giá trị cho khóa chính.
* @OneToOne, @OneToMany, @ManyToOne, @ManyToMany: Xác định quan hệ giữa các thực thể.

Hibernate là một framework mã nguồn mở phổ biến, hỗ trợ **ORM (Object-Relational Mapping)** trong các ứng dụng Java, giúp ánh xạ các đối tượng Java với các bảng trong cơ sở dữ liệu một cách tự động. Điều này có nghĩa là thay vì phải viết mã SQL trực tiếp để tương tác với cơ sở dữ liệu, các lập trình viên có thể làm việc với các đối tượng Java, còn Hibernate sẽ chuyển đổi các thao tác với đối tượng thành các câu lệnh SQL phù hợp.

**Tổng quan về Hibernate**

Hibernate được sử dụng rộng rãi trong các ứng dụng Java EE nhờ vào khả năng giảm thiểu mã SQL thủ công và tăng tính hiệu quả trong quản lý dữ liệu. Hibernate tự động chuyển đổi các đối tượng Java thành dữ liệu có thể lưu trữ trong các bảng cơ sở dữ liệu và ngược lại.

Hibernate có thể hoạt động với hầu hết các hệ quản trị cơ sở dữ liệu phổ biến như MySQL, PostgreSQL, Oracle, và SQL Server, và thậm chí còn hỗ trợ các hệ quản trị phi quan hệ như MongoDB.

**Những điểm nổi bật của Hibernate:**

* **Tính di động**: Vì Hibernate độc lập với nền tảng, nên ứng dụng có thể chuyển đổi dễ dàng giữa các loại cơ sở dữ liệu mà không cần thay đổi nhiều mã nguồn.
* **Quản lý vòng đời của đối tượng**: Hibernate giúp quản lý trạng thái của đối tượng trong suốt vòng đời của nó, từ lúc được tạo ra đến khi bị xóa.
* **Lazy Loading và Eager Loading**: Hibernate cung cấp cơ chế tải dữ liệu linh hoạt, giúp quản lý tài nguyên tốt hơn bằng cách chỉ tải dữ liệu khi cần thiết (Lazy Loading) hoặc tải toàn bộ dữ liệu ngay lập tức (Eager Loading).

**Cấu trúc và Cách hoạt động của Hibernate**

**Hibernate Configuration File (hibernate.cfg.xml):** Đây là file cấu hình chính của Hibernate, nơi lưu trữ các thông tin cấu hình về cơ sở dữ liệu (URL, tên người dùng, mật khẩu), cũng như các thuộc tính tùy chỉnh khác của Hibernate.

**Mapping File (hbm.xml):** Hibernate hỗ trợ ánh xạ giữa lớp Java và bảng cơ sở dữ liệu thông qua các file .hbm.xml, hoặc bằng cách sử dụng các annotation. Các file này định nghĩa cách Hibernate sẽ ánh xạ các thuộc tính của lớp Java sang các cột của bảng cơ sở dữ liệu.

**Session và SessionFactory:** Session là đối tượng làm việc chính của Hibernate, chịu trách nhiệm quản lý các thao tác CRUD (Create, Read, Update, Delete). SessionFactory là đối tượng chính để khởi tạo và cấu hình các phiên làm việc, quản lý các phiên kết nối với cơ sở dữ liệu. SessionFactory thường được tạo một lần khi ứng dụng khởi chạy và cung cấp các Session theo yêu cầu.

**Các Chức năng Chính của Hibernate**

**CRUD (Create, Read, Update, Delete):** Hibernate cung cấp các phương thức trực tiếp để tạo, đọc, cập nhật và xóa các đối tượng trong cơ sở dữ liệu mà không cần viết câu lệnh SQL thủ công.

**HQL (Hibernate Query Language):** HQL là ngôn ngữ truy vấn của Hibernate, gần giống với SQL nhưng sử dụng các tên lớp và thuộc tính Java thay vì các tên bảng và cột. Điều này làm cho HQL có tính di động cao hơn vì nó không phụ thuộc vào hệ quản trị cơ sở dữ liệu cụ thể.

**Caching (Bộ nhớ đệm):** Hibernate hỗ trợ cơ chế bộ nhớ đệm (Caching) để tăng hiệu suất bằng cách lưu trữ các bản ghi đã truy xuất. Hibernate có hai cấp độ bộ nhớ đệm:

* **First-level Cache**: Bộ nhớ đệm cấp đầu tiên, được bật mặc định, áp dụng cho từng phiên làm việc (Session).
* **Second-level Cache**: Bộ nhớ đệm cấp thứ hai, cho phép chia sẻ dữ liệu giữa các phiên làm việc khác nhau. Cần cài đặt thêm các thư viện như Ehcache, Infinispan để sử dụng chức năng này.

**Transaction Management (Quản lý giao dịch):** Hibernate hỗ trợ quản lý giao dịch, đảm bảo rằng các thao tác với cơ sở dữ liệu là nguyên tử, tức là nếu một phần của giao dịch thất bại, toàn bộ giao dịch sẽ bị hủy bỏ để giữ tính toàn vẹn của dữ liệu.

**Các Annotation trong Hibernate**

Ngoài cách ánh xạ với file XML, Hibernate cũng hỗ trợ ánh xạ với annotation, đơn giản hóa quá trình ánh xạ và làm cho mã dễ đọc hơn. Một số annotation phổ biến:

* @Entity: Đánh dấu một lớp là thực thể Hibernate (một bảng trong cơ sở dữ liệu).
* @Table: Xác định tên bảng mà thực thể ánh xạ tới.
* @Id: Chỉ định thuộc tính nào là khóa chính.
* @GeneratedValue: Đặt chế độ tự động tạo giá trị cho khóa chính.
* @OneToOne, @OneToMany, @ManyToOne, @ManyToMany: Xác định quan hệ giữa các thực thể.

**Lợi ích của Hibernate so với JDBC**

* **Giảm thiểu mã nguồn SQL thủ công**: Hibernate tự động hóa các thao tác CRUD, giúp giảm đáng kể mã nguồn SQL cần viết.
* **Tính di động cao**: Chuyển đổi giữa các hệ quản trị cơ sở dữ liệu dễ dàng.
* **Quản lý vòng đời đối tượng và bộ nhớ đệm**: Hibernate tối ưu hóa việc quản lý đối tượng và tăng hiệu suất nhờ cơ chế bộ nhớ đệm hai cấp.
* **Hỗ trợ các quan hệ phức tạp**: Hibernate dễ dàng xử lý các quan hệ phức tạp giữa các bảng, giảm sự phức tạp trong mã nguồn.

Tham khảo:

Java Servlet cơ bản:

<https://drive.google.com/file/d/17775vmRMHVEBhK2XNENe7_8RRPGfx0iA/view?usp=sharing>

Hibernate:

<https://viblo.asia/p/nhat-ky-hoc-hibernate-tu-so-0-7ymJXGQa4kq>

Listener:

<https://viblo.asia/p/gioi-thieu-ve-listener-trong-java-servlet-y37LdY3RVov>

# Phần II: Khảo sát, phân tích và thiết kế hệ thống

## 1. Giới thiệu về Website

Website là nền tảng hỗ trợ cho việc mua sắm chung, giúp nhóm người dùng có thể dễ dàng tổ chức các phiên đi siêu thị cùng nhau. Website cung cấp các công cụ để tạo danh sách sản phẩm, lưu trữ và cập nhật hóa đơn, tính toán chi phí và phân chia nợ một cách minh bạch. Nhờ đó, người dùng có thể tiết kiệm thời gian và công sức khi lập kế hoạch mua sắm, đảm bảo sự chính xác trong chia sẻ chi phí giữa các thành viên.

## 2. Đánh giá

Website được thiết kế với giao diện thân thiện, dễ sử dụng và trực quan, giúp người dùng dễ dàng theo dõi và cập nhật trạng thái các mặt hàng, chi phí mua sắm. Tính năng lưu trữ và so sánh hóa đơn giúp đảm bảo tính minh bạch và giảm thiểu sai sót khi tính toán nợ. Người dùng đánh giá cao về tính năng đồng bộ thông báo, cho phép mọi người trong nhóm luôn nắm được các cập nhật mới nhất. Các tính năng bảo mật giúp tạo ra không gian an toàn cho các giao dịch và thông tin cá nhân của người dùng.

## 3. Mục đích

Website ra đời nhằm giải quyết các bất tiện trong việc chia sẻ chi phí mua sắm nhóm, giúp mọi người có thể phân chia hóa đơn một cách công bằng và dễ dàng. Ngoài ra, hệ thống quản lý hóa đơn, giỏ hàng và thông báo còn tạo ra một phương thức quản lý mua sắm hiệu quả cho các nhóm, như gia đình, nhóm bạn hay đồng nghiệp. Mục tiêu cuối cùng là tạo ra một nền tảng hỗ trợ người dùng trong quá trình đi mua sắm chung, tiết kiệm thời gian và tránh nhầm lẫn trong chia sẻ chi phí.

## 4, Giải pháp xây dựng hệ thống

* **Chức năng chính**:
  + **Quản lý sản phẩm và danh mục**:
    - Hiển thị danh sách các sản phẩm theo từng danh mục.
    - Cho phép thêm, sửa, và cập nhật các mặt hàng (giá cả, ghi chú).
  + **Giỏ hàng và phiên đi siêu thị**:
    - Thêm sản phẩm vào giỏ hàng và lưu trữ trong Session.
    - Tạo và quản lý phiên đi siêu thị; gửi lời mời cho người dùng khác tham gia phiên mua sắm.
    - Cập nhật trạng thái phiên và các sản phẩm sau mỗi lần đi siêu thị.
  + **Chia sẻ hóa đơn và thanh toán**:
    - Tính toán hóa đơn và chia đều chi phí sau khi hoàn tất mua sắm.
    - Ghi nợ và quản lý các khoản chi trả của từng người tham gia.
  + **Thông báo và tương tác**:
    - Hiển thị thông báo cập nhật từ các thành viên trong phiên mua sắm.
  + **Quản lý tài khoản**:
    - Đăng ký, đăng nhập, đăng xuất, quên mật khẩu, và lưu đăng nhập (remember me).
  + **Chức năng quản trị** (Admin):
    - Quản lý người dùng và thông tin tài khoản.
    - Quản lý sản phẩm, giỏ hàng, và các phiên đi siêu thị.
* **Yêu cầu phi chức năng**:
  + **Giao diện hiện đại**:
    - Giao diện thân thiện, dễ sử dụng, cập nhật theo xu hướng thiết kế mới nhất.
  + **Hiệu suất cao**:
    - Hệ thống tối ưu để xử lý lượng lớn người dùng đồng thời.
  + **Bảo mật thông tin**:
    - Đảm bảo an toàn cho các thông tin cá nhân và giao dịch người dùng.
  + **Tính năng tìm kiếm nhanh chóng**:
    - Tìm kiếm mặt hàng và sản phẩm nhanh và chính xác.
  + **Cập nhật liên tục**:
    - Cập nhật trạng thái sản phẩm, giỏ hàng, và phiên đi siêu thị ngay tức thì.

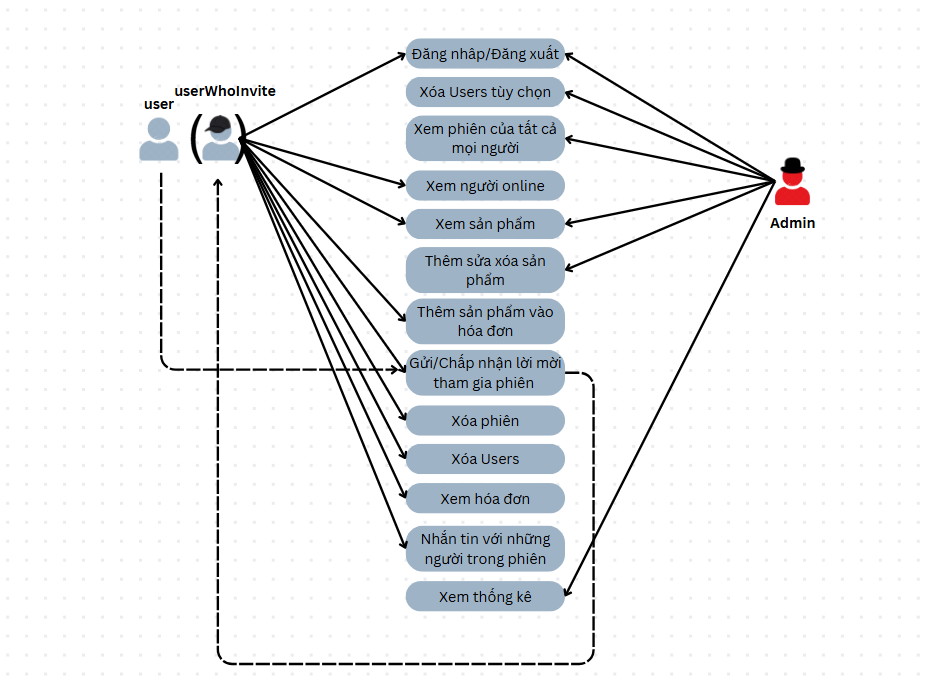
## 5. Phân tích và thiết kế hệ thống

**1. Yêu cầu chi tiết về website**

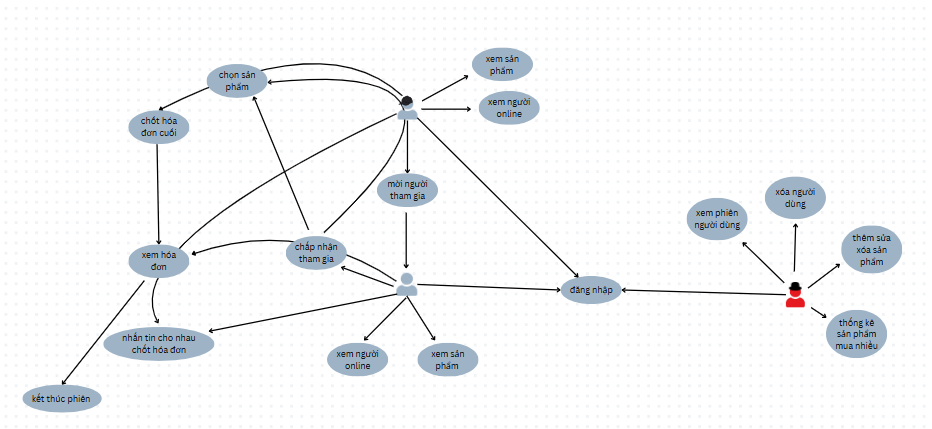
* **Giao diện hiển thị**:
  + **Danh mục sản phẩm**:
    - Hiển thị các danh mục mặt hàng và từng sản phẩm trong từng mục.
  + **Chi tiết sản phẩm**:
    - Cung cấp thông tin chi tiết của từng sản phẩm, bao gồm giá, tên, và ghi chú nếu có.
* **Tính năng người dùng**:
  + **Quản lý tài khoản**:
    - Đăng ký, đăng nhập, chỉnh sửa thông tin cá nhân.
  + **Sử dụng giỏ hàng**:
    - Thêm sản phẩm vào giỏ hàng chung, lưu Session và quản lý giỏ hàng dễ dàng.
  + **Tham gia và quản lý phiên đi siêu thị**:
    - Tạo phiên đi siêu thị mới, gửi lời mời cho người dùng khác.
    - Cập nhật trạng thái phiên đi siêu thị sau mỗi lần mua sắm.
  + **Chia sẻ và tính toán hóa đơn**:
    - Xác nhận hóa đơn, tính toán chi phí và ghi nợ sau mỗi lần mua sắm.
  + **Thông báo cập nhật**:
    - Nhận thông báo về các cập nhật từ thành viên khác liên quan đến giỏ hàng và phiên đi siêu thị.
* **Tính năng quản trị viên**:
  + **Quản lý người dùng**:
    - Xem, xóa và thông tin tài khoản của người dùng.
  + **Quản lý sản phẩm và danh mục**:
    - Kiểm soát danh mục sản phẩm và mặt hàng chi tiết.
  + **Quản lý phiên đi siêu thị**:
    - Kiểm soát thông tin các phiên đi siêu thị, thống kê các khoản chi tiêu của từng phiên.

## 6, Biểu đồ

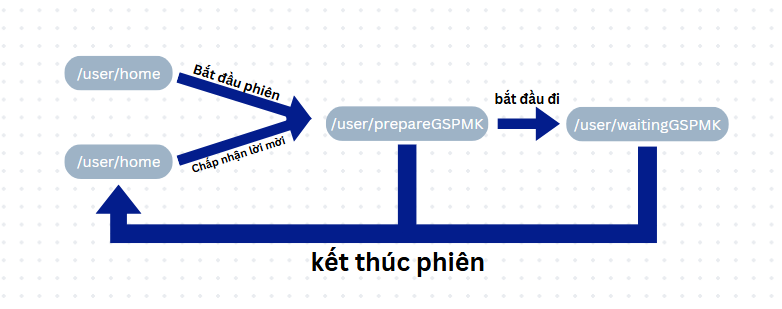
1, Biểu đồ Usecase tổng quát



Biểu đồ Usecase phân rã

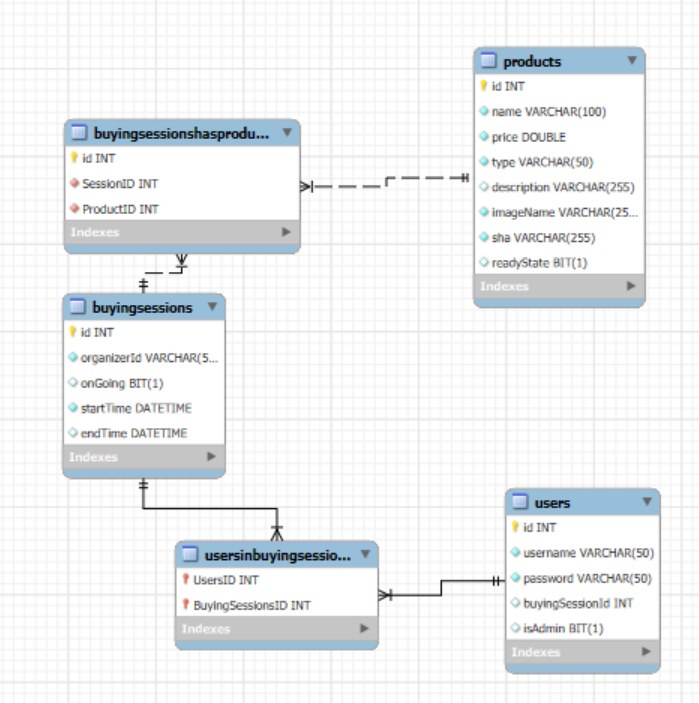


Luồng đi trong trang web



# Phần III: Xây dựng trang Web

## Database



**1. Bảng users**

* **id**: Khóa chính, định danh duy nhất của mỗi người dùng.
* **username**: Tên đăng nhập của người dùng (kiểu VARCHAR, tối đa 50 ký tự).
* **password**: Mật khẩu của người dùng (kiểu VARCHAR, tối đa 50 ký tự).
* **buyingSessionId**: Khóa ngoại, liên kết đến một phiên mua hàng trong bảng buyingsessions.
* **isAdmin**: Xác định người dùng có phải là quản trị viên hay không (kiểu BIT, 1 nghĩa là có, 0 nghĩa là không).

**2. Bảng products**

* **id**: Khóa chính, định danh duy nhất của mỗi sản phẩm.
* **name**: Tên của sản phẩm (kiểu VARCHAR, tối đa 100 ký tự).
* **price**: Giá của sản phẩm (kiểu DOUBLE).
* **type**: Loại sản phẩm (kiểu VARCHAR, tối đa 50 ký tự).
* **description**: Mô tả về sản phẩm (kiểu VARCHAR, tối đa 255 ký tự).
* **imageName**: Tên tệp ảnh của sản phẩm (kiểu VARCHAR, tối đa 255 ký tự).
* **sha**: Chuỗi hash SHA để xác minh sản phẩm (kiểu VARCHAR, tối đa 255 ký tự).
* **readyState**: Trạng thái sẵn sàng của sản phẩm (kiểu BIT, 1 là sẵn sàng, 0 là chưa sẵn sàng).

**3. Bảng buyingsessions**

* **id**: Khóa chính, định danh duy nhất của mỗi phiên mua hàng.
* **organizerId**: ID của người tổ chức phiên mua hàng (kiểu VARCHAR, tối đa 50 ký tự).
* **onGoing**: Trạng thái phiên mua hàng (kiểu BIT, 1 là đang diễn ra, 0 là đã kết thúc).
* **startTime**: Thời gian bắt đầu của phiên mua hàng (kiểu DATETIME).
* **endTime**: Thời gian kết thúc của phiên mua hàng (kiểu DATETIME).

**4. Bảng buyingsessionshasproducts**

* **id**: Khóa chính, định danh duy nhất của bảng.
* **SessionID**: Khóa ngoại, liên kết đến một phiên mua hàng trong bảng buyingsessions.
* **ProductID**: Khóa ngoại, liên kết đến một sản phẩm trong bảng products.

Bảng này là bảng trung gian, được sử dụng để xác định sản phẩm nào có trong mỗi phiên mua hàng (mối quan hệ nhiều-nhiều giữa buyingsessions và products).

**5. Bảng usersinbuyingsessions**

* **UsersID**: Khóa ngoại, liên kết đến người dùng trong bảng users.
* **BuyingSessionsID**: Khóa ngoại, liên kết đến phiên mua hàng trong bảng buyingsessions.

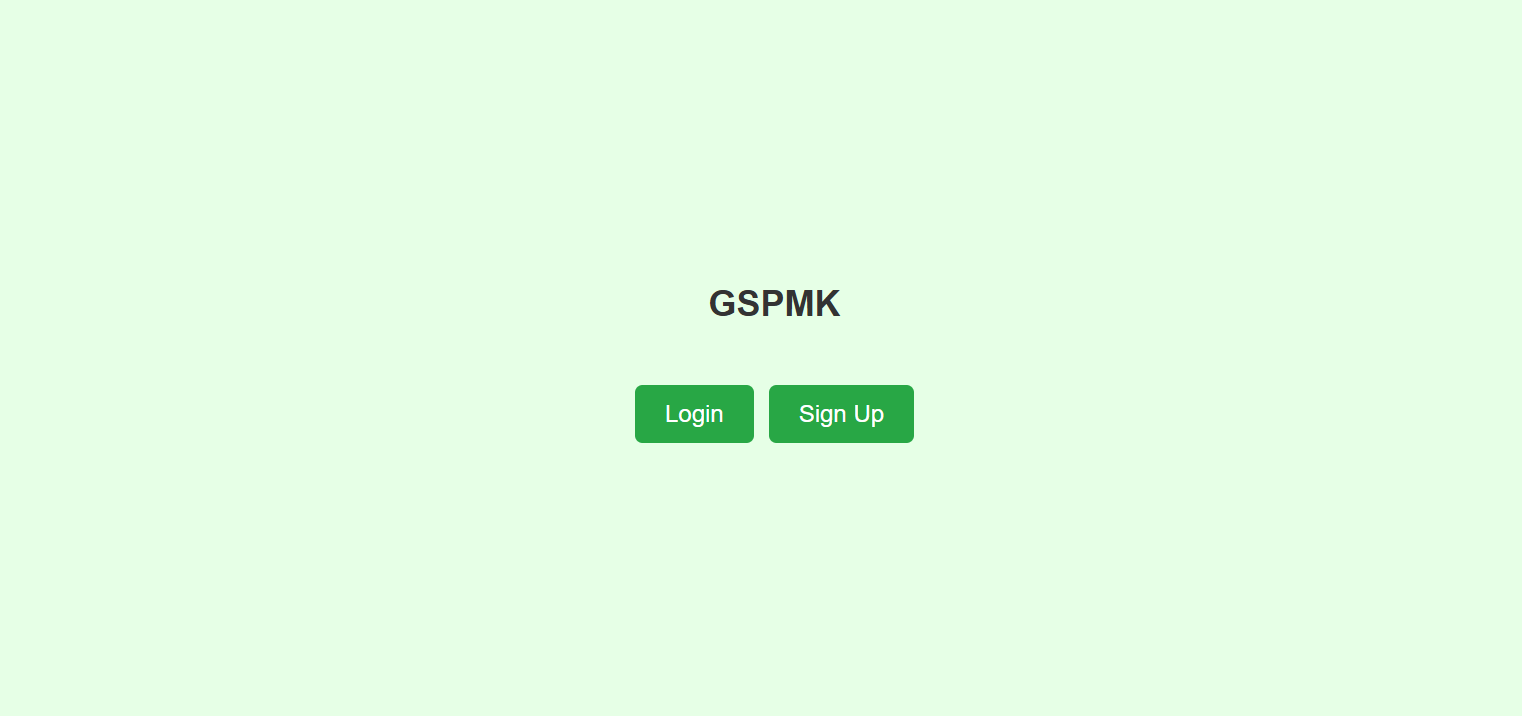
Bảng này là bảng trung gian, xác định người dùng nào tham gia vào các phiên mua hàng (mối quan hệ nhiều-nhiều giữa users và buyingsessions).

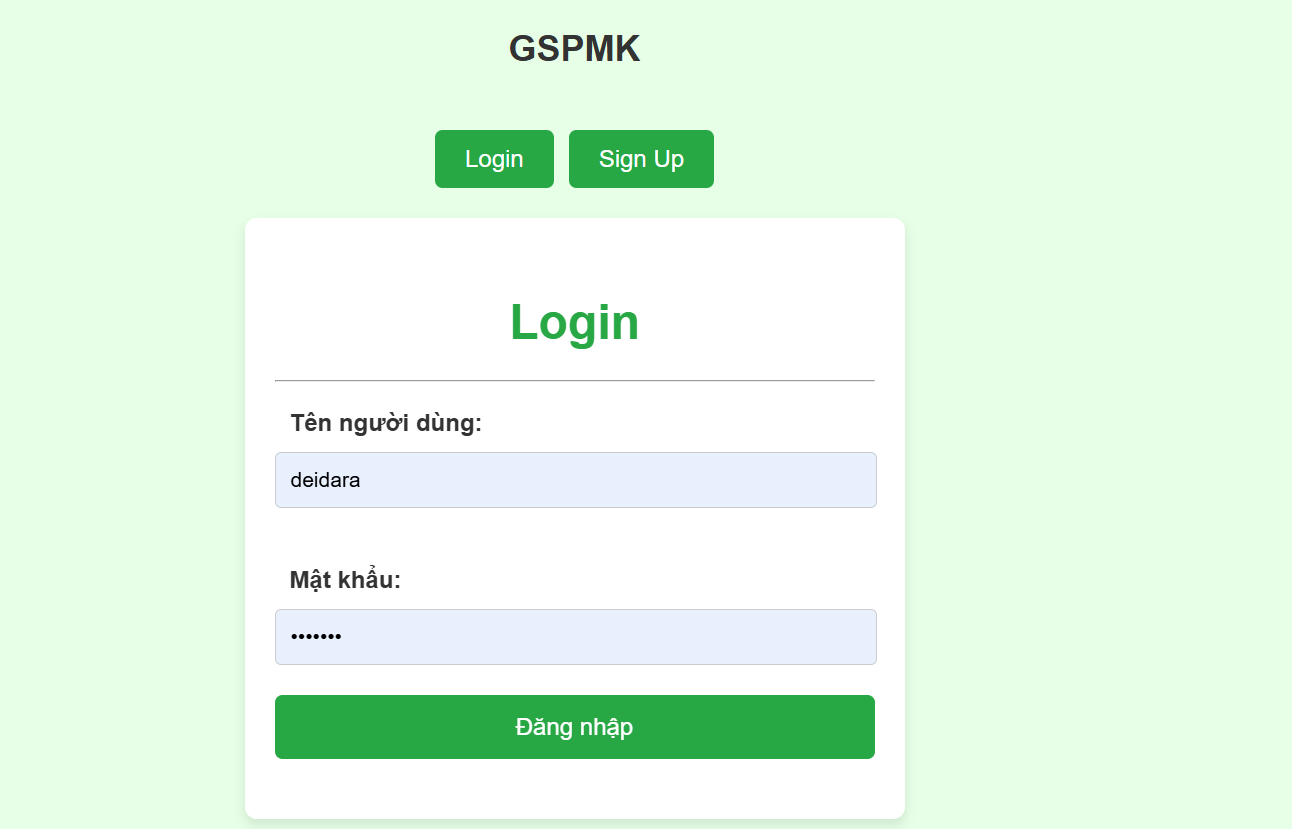
**Tóm tắt mối quan hệ**

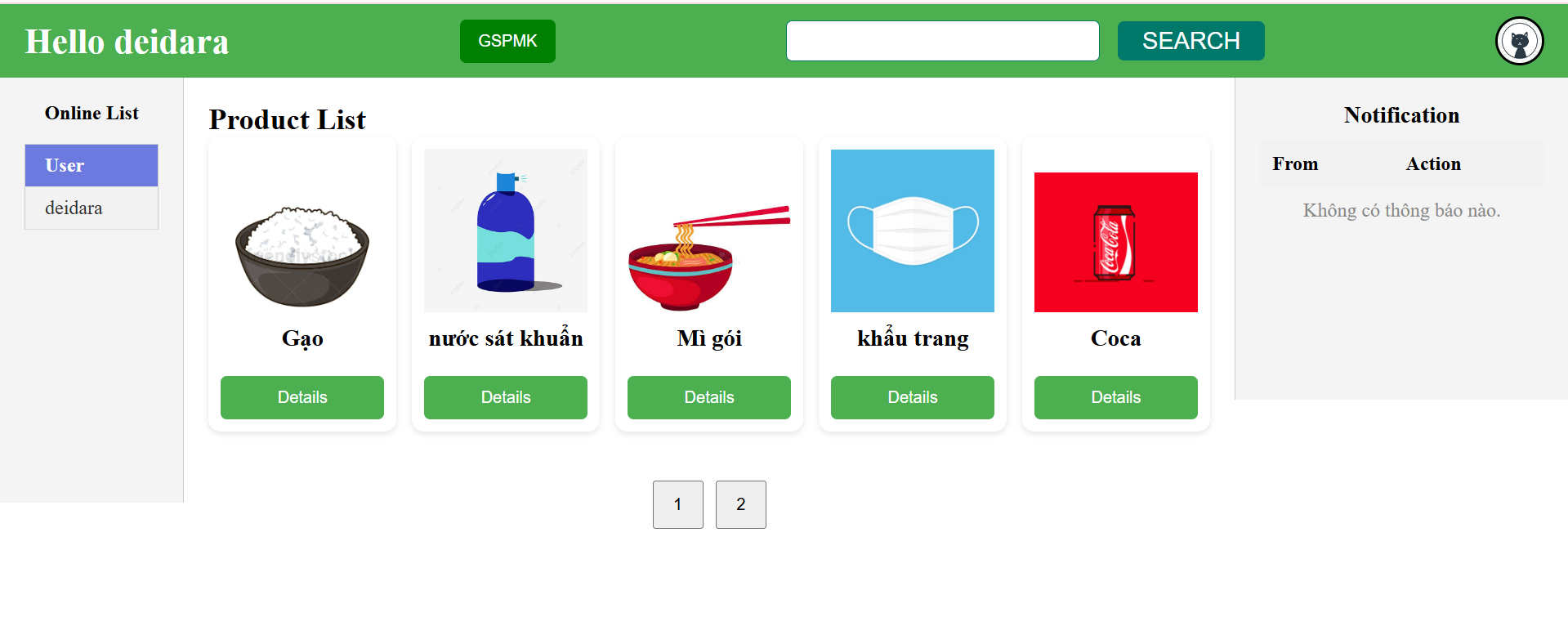
* users có mối quan hệ một-nhiều với buyingsessions.
* products có mối quan hệ nhiều-nhiều với buyingsessions thông qua bảng buyingsessionshasproducts.
* users có mối quan hệ nhiều-nhiều với buyingsessions thông qua bảng usersinbuyingsessions.

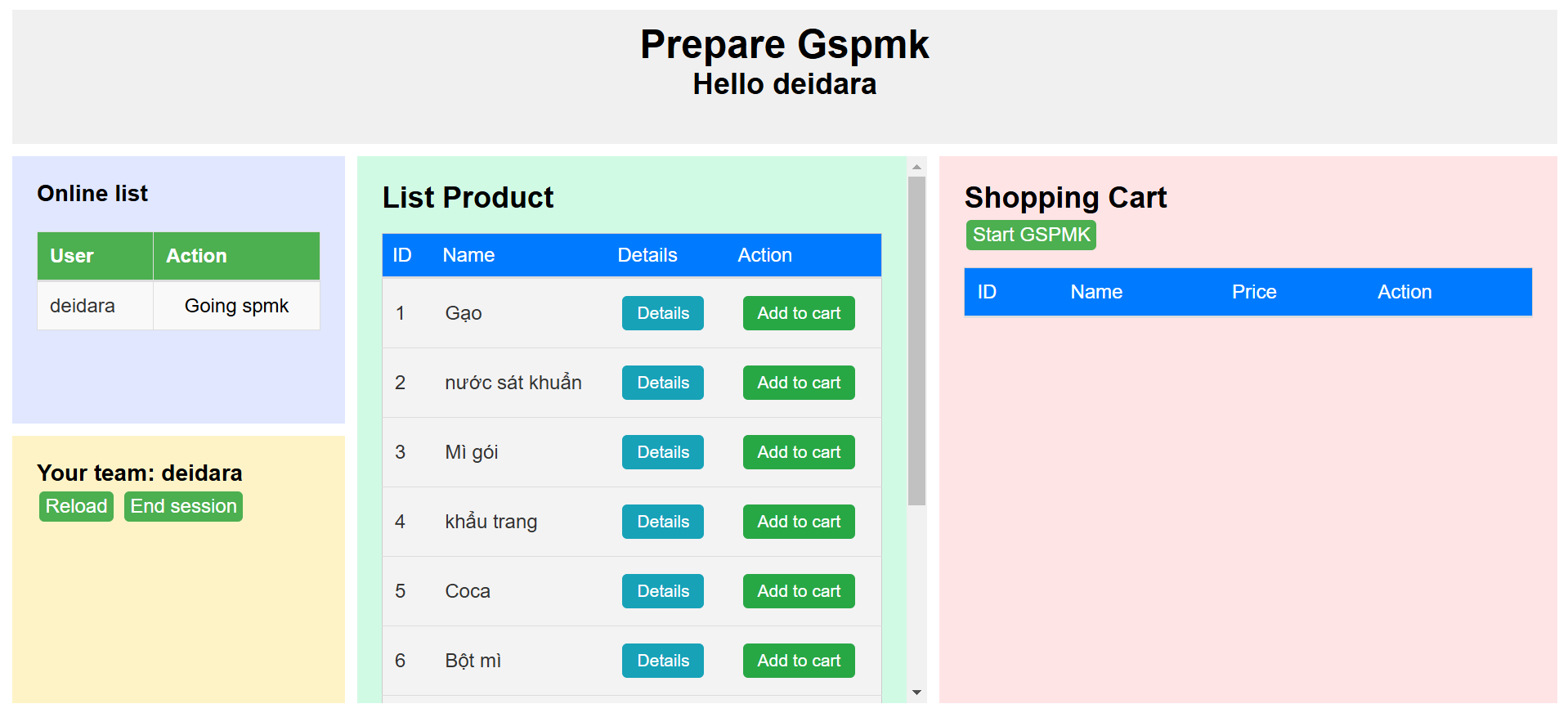
## Trang Web

Trang welcome

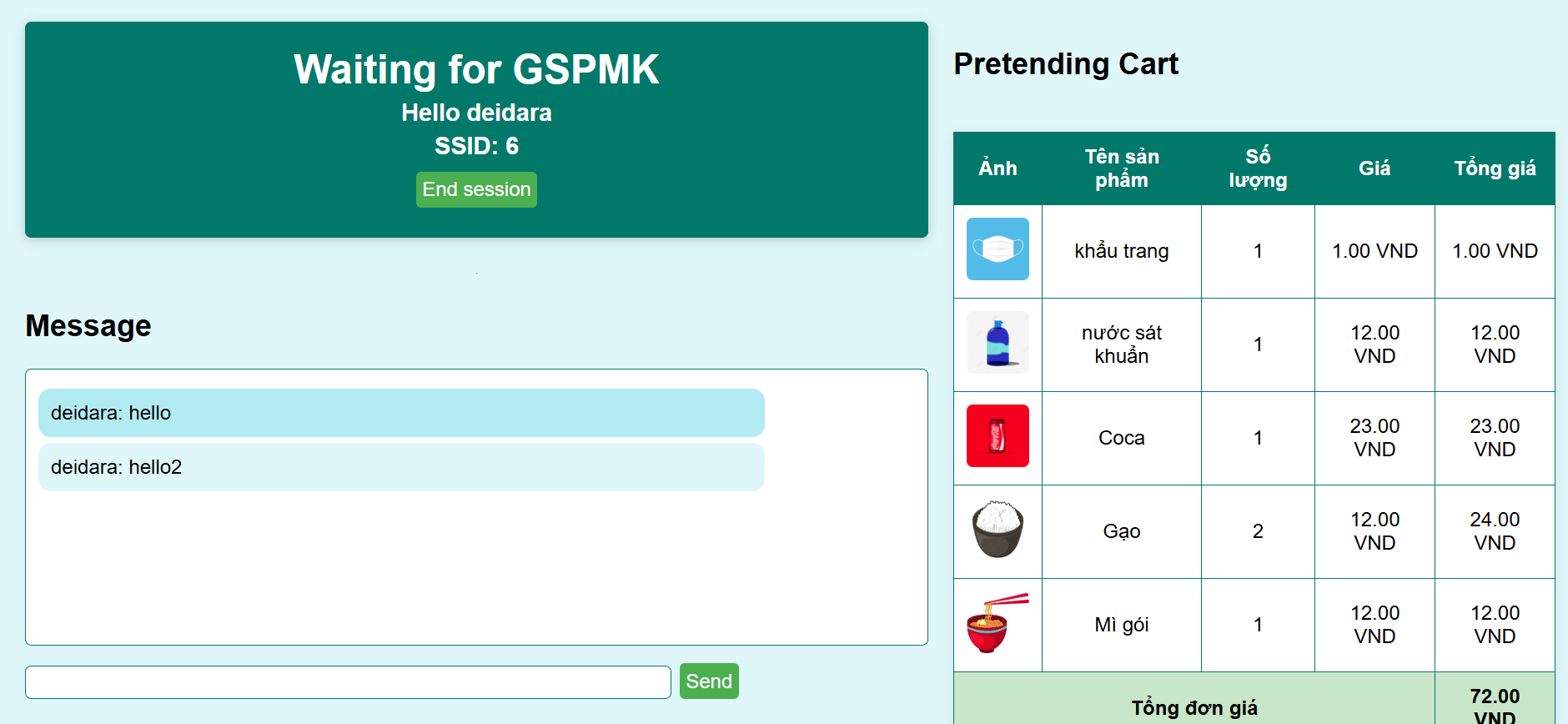


Trang đăng nhập:  


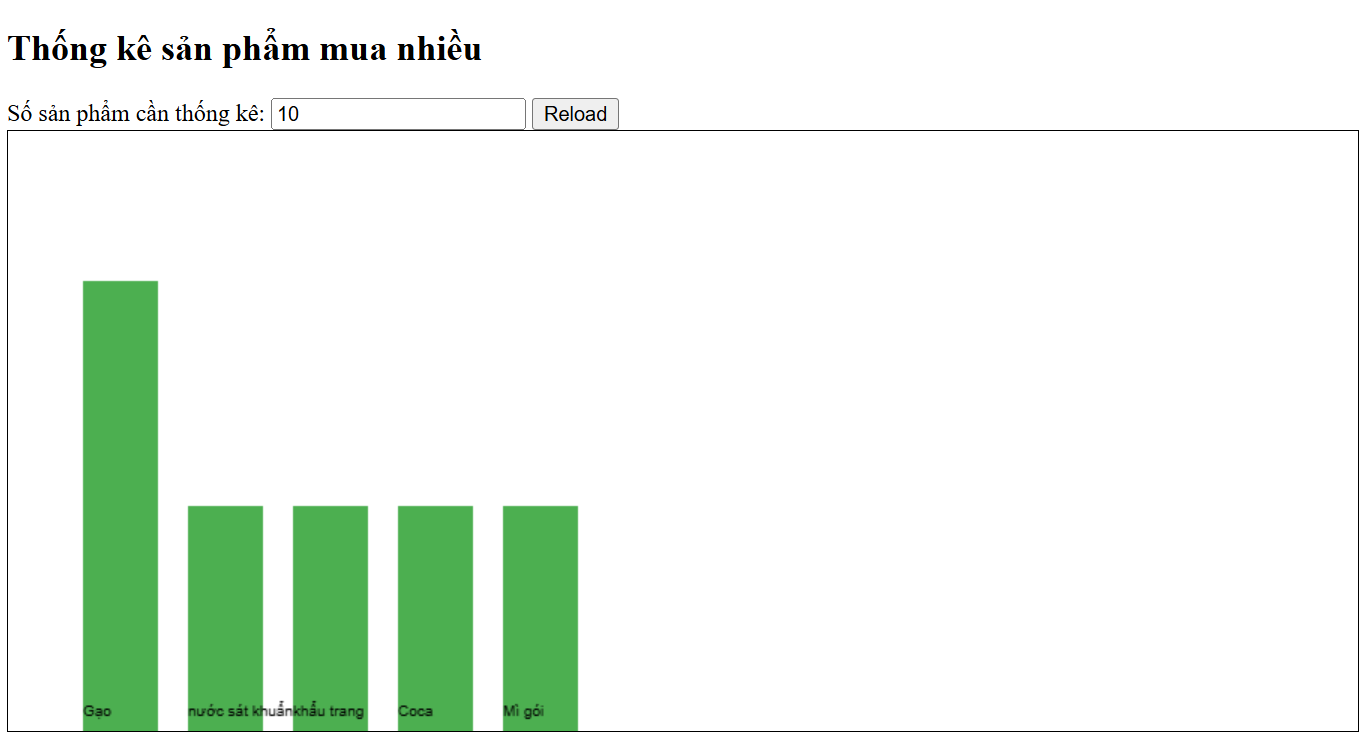
Trang chủ:  


Trong phiên đi mua sắm:  


Kết thúc lượt đi và trang share bill:



Thống kê (admin):



# Phần IV: Kết luận

Dự án **“Quản lý Mua Sắm và Chia Sẻ Hóa Đơn Nhóm”** đã đạt được những kết quả tích cực trong việc xây dựng một nền tảng hỗ trợ quản lý mua sắm nhóm. Hệ thống không chỉ giúp đơn giản hóa quy trình lập danh sách, tạo và quản lý phiên đi siêu thị mà còn hỗ trợ chia sẻ chi phí một cách minh bạch giữa các thành viên, tạo điều kiện để tăng cường tính cộng đồng và giảm thiểu các vấn đề trong việc phân chia hóa đơn.

**Kết quả đạt được:**

* **Giao diện trực quan và thân thiện**: Hệ thống cung cấp một giao diện dễ sử dụng cho cả người dùng và quản trị viên, hỗ trợ người dùng quản lý giỏ hàng, phiên mua sắm và hóa đơn một cách dễ dàng.
* **Tính năng tạo phiên đi siêu thị và chia sẻ chi phí hiệu quả**: Tính năng này giúp đơn giản hóa quy trình quản lý mua sắm nhóm, đồng thời tạo điều kiện thuận lợi cho việc chia sẻ chi phí minh bạch, góp phần tăng cường sự gắn kết và hỗ trợ giữa các thành viên.
* **Bảo mật và hiệu suất cơ bản**: Hệ thống đảm bảo bảo mật thông tin cá nhân và giao dịch của người dùng ở mức cơ bản, đồng thời xử lý tốt các thao tác như lưu trữ giỏ hàng và hóa đơn, hỗ trợ việc quản lý và phân chia chi phí.

**Hạn chế:**

* **Chức năng còn chưa hoàn chỉnh**: Một số tính năng quản lý sản phẩm, phiên đi siêu thị và hóa đơn vẫn còn cơ bản, cần hoàn thiện để đáp ứng tốt hơn nhu cầu thực tế của người dùng.
* **Chưa tối ưu hóa mã nguồn**: Hệ thống hiện tại chưa được tối ưu hóa đầy đủ về hiệu suất, dẫn đến tốc độ xử lý và trải nghiệm người dùng có thể bị ảnh hưởng khi có lượng truy cập lớn.

**Phương hướng khắc phục và phát triển:**

* **Hoàn thiện các tính năng cơ bản**: Nâng cấp và bổ sung các tính năng hiện có như quản lý giỏ hàng, hóa đơn, và thông báo cập nhật để đáp ứng tốt hơn nhu cầu của người dùng.
* **Tối ưu hóa mã nguồn và hiệu suất**: Tiến hành tối ưu hóa mã nguồn nhằm cải thiện hiệu suất, đảm bảo hệ thống hoạt động ổn định và nhanh chóng hơn.
* **Cải thiện trải nghiệm người dùng**: Tập trung vào tối ưu hóa giao diện, quy trình mua sắm và thanh toán, đảm bảo người dùng có trải nghiệm liền mạch và thuận tiện.
* **Phát triển các tính năng hỗ trợ kết nối cộng đồng**: Đưa vào các tính năng để tăng cường khả năng tương tác giữa các thành viên trong phiên đi siêu thị, hỗ trợ tốt hơn việc chia sẻ thông tin, góp phần nâng cao tính cộng đồng và gắn kết.