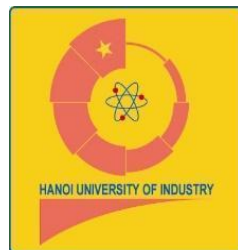


TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI  
KHOA CÔNG NGHỆ THÔNG TIN

=====\*\*\*=====



BÁO CÁO BÀI TẬP LỚN HỌC PHẦN:  
LẬP TRÌNH JAVA  
XÂY DỰNG PHẦN MỀM QUẢN LÝ  
TIỆM TẠP HOÁ XANH

GVHD: TS. HÀ MẠNH ĐÀO  
Nhóm - Lớp: 15 - 20241IT6019001  
Thành viên: Nguyễn Ngọc Huy - 2021601338  
Lê Thị Thanh Thảo - 2021601777  
Nguyễn Đình Thường - 2022607447

Hà Nội – 2024

## LỜI MỞ ĐẦU

Trong bối cảnh kinh tế phát triển mạnh mẽ, nhu cầu mua sắm các mặt hàng thiết yếu của người dân ngày càng tăng cao. Việc quản lý hiệu quả một tiệm tạp hóa, bao gồm các khâu như nhập hàng, bán hàng, quản lý tồn kho, và chăm sóc khách hàng đóng một vai trò vô cùng quan trọng.

Chính vì thế, việc ứng dụng công nghệ thông tin trong quản lý tiệm tạp hóa đã trở thành một giải pháp thiết yếu, giúp nâng cao năng suất, tối ưu hóa quy trình vận hành, và cải thiện chất lượng dịch vụ. Đặc biệt, một phần mềm quản lý hiện đại sẽ hỗ trợ tiệm tạp hóa trong việc quản lý hàng hóa, hóa đơn, và doanh thu một cách khoa học, chính xác và tiện lợi.

Xuất phát từ nhu cầu thực tiễn này, đề tài "Xây dựng phần mềm quản lý tiệm tạp hóa Xanh bằng Java Swing" được thực hiện với mục tiêu cung cấp một giải pháp quản lý toàn diện, hiệu quả và dễ sử dụng. Phần mềm sẽ hỗ trợ các chức năng chính như quản lý thông tin hàng hóa, theo dõi doanh thu, quản lý tồn kho,... Bằng việc sử dụng ngôn ngữ lập trình Java kết hợp với thư viện giao diện đồ họa Swing - một công nghệ mạnh mẽ và linh hoạt - hướng tới việc tạo ra một hệ thống thân thiện, trực quan và đáp ứng đầy đủ các yêu cầu nghiệp vụ của tiệm tạp hóa.

Dù đã cố gắng hoàn thiện và đầy đủ, chúng em nhận thức rằng báo cáo vẫn có thể mắc phải một số sai sót. Vì vậy, chúng em rất mong nhận được sự góp ý từ quý thầy cô và các bạn để bản báo cáo có thể được hoàn thiện và nâng cao hơn.

Nhóm xin chân thành cảm ơn!

*Nhóm sinh viên thực hiện*

*Nhóm 15*

## MỤC LỤC

<b>LỜI MỞ ĐẦU .....</b>	<b>2</b>
<b>DANH MỤC HÌNH ẢNH.....</b>	<b>5</b>
<b>DANH MỤC BẢNG BIỂU .....</b>	<b>6</b>
<b>DANH MỤC VĂN TẮT.....</b>	<b>7</b>
<b>PHẦN 1: MỞ ĐẦU.....</b>	<b>8</b>
1.1 Giới thiệu đề tài.....	8
1.2 Xác định nội dung đề tài .....	8
1.3 Xác định kiến thức và các công cụ thực hiện đề tài .....	9
<b>PHẦN 2: NỘI DUNG.....</b>	<b>12</b>
<b>CHƯƠNG 1: KHẢO SÁT HỆ THỐNG .....</b>	<b>12</b>
1.1 Tài liệu đặc tả yêu cầu người dùng .....	12
1.1.1 Khảo sát sơ bộ .....	12
1.1.2 Tài liệu đặc tả yêu cầu người dùng .....	12
1.1.3 Mô hình hoá chức năng hệ thống.....	13
1.1.4 Mô tả chi tiết các use case.....	15
1.1.4.1 Use case Đăng nhập (Nguyễn Ngọc Huy) .....	15
1.1.4.2 Use case Đăng kí (Nguyễn Ngọc Huy).....	16
1.1.4.3 Use case Tạo hoá đơn (Lê Thị Thanh Thảo).....	17
1.1.4.4 Use case Xác thực tài khoản (Lê Thị Thanh Thảo) .....	18
1.1.4.5 Use case Quản lý sản phẩm (Nguyễn Đình Thường) .....	19
1.1.4.5 Use case Xem chi tiết hoá đơn(Nguyễn Đình Thường).....	21
1.1.5 Mô hình hoá dữ liệu của hệ thống.....	22
<b>CHƯƠNG 2: THIẾT KẾ HỆ THỐNG.....</b>	<b>24</b>
2.1 Thiết kế giao diện.....	24
2.1.1 Giao diện đăng nhập .....	24
2.1.2 Giao diện đăng kí .....	25
2.1.3 Giao diện quên mật khẩu .....	26
2.1.4 Giao diện màn hình chính .....	27
2.1.5 Giao diện quản lý loại sản phẩm.....	28

2.1.6 Giao diện quản lý sản phẩm .....	29
2.1.7 Giao diện quản lý hoá đơn .....	30
2.2 Thiết kế dữ liệu - Ánh xạ lớp sang bảng .....	30
Bảng User (Quản trị viên) .....	30
Bảng Category (Loại sản phẩm) .....	31
Bảng Product (Sản phẩm) .....	31
Bảng Bill (Hoá đơn) .....	31
<b>CHƯƠNG 3: CÀI ĐẶT VÀ TRIỂN KHAI .....</b>	<b>32</b>
3.1 Giới thiệu công cụ .....	32
3.2 Giới thiệu về Eclipse .....	32
<b>CHƯƠNG 4: THỰC HIỆN BÀI TOÁN .....</b>	<b>33</b>
4.1 Các tiện ích (Lớp util) .....	33
4.1.1 Làm việc với form .....	33
4.1.2 Thao tác đọc ghi file .....	34
4.1.3 Gửi Email .....	36
4.1.4 Mã hoá mật khẩu .....	38
4.1.5 Tạo mật khẩu ngẫu nhiên .....	38
4.1.6 Tạo mã hoá đơn ngẫu nhiên .....	38
4.2 Chức năng đăng nhập .....	39
4.3 Chức năng đăng kí .....	41
4.5 Màn hình chính .....	46
4.6 Chức năng quản lý loại sản phẩm .....	49
4.7 Chức năng quản lý sản phẩm .....	53
4.8 Chức năng quản lý hoá đơn .....	56
4.9 Chức năng tạo hoá đơn .....	59
4.10 Chức năng thống kê .....	61
4.11 Chức năng đổi mật khẩu .....	62
4.12 Chức năng xác minh tài khoản .....	64
<b>PHẦN 3: TỔNG KẾT .....</b>	<b>67</b>
a. Tổng kết nội dung .....	67
b. Hướng phát triển trong tương lai .....	67

**DANH MỤC HÌNH ẢNH**

Hình I: Biểu đồ use case tổng quát.....	14
Hình II: Biểu đồ phân rã use case .....	14
Hình III: Biểu diễn mối quan hệ giữa các lớp .....	23
Hình IV: Thiết kế giao diện đăng nhập .....	24
Hình V: Thiết kế giao diện đăng ký .....	25
Hình VI: Thiết kế giao diện quên mật khẩu .....	26
Hình VII: Thiết kế giao diện màn hình chính .....	27
Hình VIII: Thiết kế giao diện quản lý loại sản phẩm.....	28
Hình IX: Thiết kế giao diện quản lý sản phẩm .....	29
Hình X: Thiết kế giao diện quản lý hoá đơn .....	30
Hình XI: Giao diện đăng nhập .....	39
Hình XII: Giao diện đăng ký .....	41
Hình XIII: Giao diện quên mật khẩu.....	44
Hình XIV: Giao diện màn hình chính .....	46
Hình XV: Giao diện quản lý sản phẩm .....	49
Hình XVI: Giao diện thêm loại sản phẩm.....	50
Hình XVII: Giao diện quản lý sản phẩm .....	53
Hình XVIII: Giao diện thêm sản phẩm .....	54
Hình XIX: Giao diện quản lý hoá đơn .....	57
Hình XX: Giao diện tạo hoá đơn.....	59
Hình XXI: Giao diện thống kê .....	61
Hình XXII: Giao diện đổi mật khẩu.....	63
Hình XXIII: Giao diện xác minh tài khoản.....	65

## **DANH MỤC BẢNG BIỂU**

Bảng I: Các lớp cơ bản.....	22
Bảng II: Bảng User (Quản trị viên).....	30
Bảng III: Bảng Category (Loại sản phẩm).....	31
Bảng IV: Bảng Product (Sản phẩm).....	31
Bảng V: Bảng Bill (Hoá đơn).....	31

## DANH MỤC VĂN TẮT

# PHẦN 1: MỞ ĐẦU

## 1.1 Giới thiệu đề tài

Tên đề tài: Xây dựng phần mềm quản lý tiệm tạp hoá Xanh.

Lý do lựa chọn:

- Hiện nay, công nghệ thông tin được xem là một ngành mũi nhọn của quốc gia, đặc biệt là của các nước đang phát triển của nước ta. Sự bùng nổ thông tin và sự phát triển mạnh mẽ của công nghệ kỹ thuật số, muốn phát triển thì phải áp dụng tin học hóa vào tất cả các ngành các lĩnh vực. Cùng với sự phát triển nhanh chóng về phần cứng máy tính, các phần mềm càng trở nên đa dạng, phong phú, hoàn thiện hơn và hỗ trợ hiệu quả cho con người. Các phần mềm hiện nay ngày càng hỗ trợ cho người dùng thuận tiện sử dụng, thời gian xử lý nhanh chóng, và một số nghiệp vụ được tự động hóa cao. Các phần mềm giúp tiết kiệm một lượng lớn thời gian, công sức của con người, tăng độ chính xác và hiệu quả trong công việc.
- Nhận thức được tầm quan trọng của việc quản lý mua bán hàng, nhóm em đã chọn đề tài “Xây dựng phần mềm quản lý tiệm tạp hóa Xanh” để làm đề tài nghiên cứu. Phần mềm có nhiệm vụ quản lý loại sản phẩm, quản lý sản phẩm, quản lý tài khoản, quản lý hóa đơn, thống kê doanh thu,...

## 1.2 Xác định nội dung đề tài

- Áp dụng phương pháp lập trình hướng đối tượng trong lập trình java nhằm hoàn thiện bài tập lớn và đạt hiệu quả thực tế.
- Mô hình hoá của bài toán:
  - Quản trị viên: Sau khi đăng nhập sẽ quản lý toàn bộ hệ thống.
  - Quản lý loại sản phẩm: Xem, thêm, sửa, xóa thông tin loại sản phẩm.
  - Quản lý sản phẩm: Xem, thêm, sửa, xóa thông tin sản phẩm.
  - Quản lý hóa đơn: Tạo, xem chi tiết hóa đơn.
  - Quản lý tài khoản: Xác minh quyền truy cập cho người dùng.
  - Thống kê doanh thu: Thống kê doanh thu theo từng tháng trong năm.
- Ý nghĩa thực tiễn của đề tài:
  - Giúp chủ tiệm quản lý hiệu quả thông tin hàng hóa, khách hàng, và doanh thu. Hỗ trợ theo dõi tồn kho chính xác, tránh tình trạng thiếu hoặc dư thừa hàng hóa.



- Giảm thiểu thời gian và công sức trong việc ghi chép và tìm kiếm thông tin. Tự động hóa các tác vụ quản lý như lập hóa đơn, báo cáo doanh thu và nhập xuất hàng.
- Đáp ứng nhanh chóng các yêu cầu của khách hàng nhờ khả năng truy xuất dữ liệu nhanh và chính xác. Tăng sự hài lòng của khách hàng với quy trình phục vụ chuyên nghiệp.

### 1.3 Xác định kiến thức và các công cụ thực hiện đề tài

#### 1.3.1 Kiến thức học tập

Ngôn ngữ Java cơ bản:

- + Cấu trúc chương trình Java
- + Kiểu dữ liệu và chuyển đổi kiểu dữ liệu
- + Các toán tử
- + Cấu trúc điều khiển

Xử lý dữ liệu và đối tượng:

- + Mảng và xử lý mảng
- + Lớp, đối tượng, và các hàm khởi tạo
- + Phương thức tĩnh (static)
- + Mảng đối tượng
- + Nạp chồng phương thức

Lập trình hướng đối tượng:

- + Kết tập, kế thừa
- + Tính trừu tượng, đa hình và interface
- + Ghi đè phương thức

Xử lý nâng cao:

- + Xử lý ngoại lệ
- + I/O theo luồng và thao tác với tệp
- + Cấu trúc Collection

Phát triển giao diện: Giao diện Java Swing

#### 1.3.2 Công cụ thực hiện đề tài

Eclipse

Eclipse là một môi trường phát triển tích hợp (IDE) mã nguồn mở nổi bật, được sử dụng rộng rãi trong cộng đồng lập trình viên trên toàn thế giới. Ra đời vào năm 2001 bởi IBM và hiện được quản lý bởi Eclipse Foundation, Eclipse nhanh

chóng trở thành công cụ phát triển hàng đầu nhờ tính linh hoạt và khả năng mở rộng mạnh mẽ. Ban đầu được thiết kế chủ yếu để hỗ trợ lập trình Java, Eclipse đã không ngừng phát triển để tích hợp nhiều ngôn ngữ lập trình khác nhau như C/C++, Python, PHP, và nhiều ngôn ngữ khác thông qua hệ thống plugin phong phú.

Một trong những điểm mạnh của Eclipse là khả năng hoạt động đa nền tảng, cho phép chạy trên các hệ điều hành phổ biến như Windows, macOS, và Linux. Giao diện của Eclipse được thiết kế thân thiện, trực quan, hỗ trợ tốt cho cả người mới học lập trình và các lập trình viên chuyên nghiệp. Các tính năng nổi bật của Eclipse bao gồm gợi ý mã thông minh, làm nổi bật cú pháp, tự động hoàn thành mã, và hệ thống gỡ lỗi mạnh mẽ, giúp tăng hiệu suất làm việc đáng kể. Ngoài ra, khả năng tích hợp với Git và các công cụ quản lý phiên bản khác cũng là một lợi thế lớn, đặc biệt đối với các dự án nhóm hoặc dự án lớn.

Các chức năng quan trọng của eclipse

- + Hỗ trợ phát triển Java toàn diện: Eclipse cung cấp công cụ mạnh mẽ để phát triển các ứng dụng Java từ desktop (Java SE), web (Java EE), đến giao diện đồ họa nâng cao (JavaFX).
- + Hỗ trợ đa ngôn ngữ lập trình: Bên cạnh Java, Eclipse còn hỗ trợ các ngôn ngữ khác như PHP, C/C++, HTML5, JavaScript, Groovy và nhiều ngôn ngữ khác thông qua hệ thống plugin phong phú.
- + Giao diện người dùng trực quan: Với thiết kế thân thiện và dễ sử dụng, Eclipse giúp lập trình viên làm việc hiệu quả hơn nhờ các công cụ kéo-thả hỗ trợ phát triển ứng dụng nhanh chóng.
- + Quản lý dự án hiệu quả: Eclipse tích hợp các công cụ quản lý dự án, hỗ trợ kiểm soát phiên bản như Git và SVN, đồng thời làm việc tốt với các hệ thống xây dựng phổ biến như Maven và Ant.
- + Trình biên tập mã thông minh: IDE này cung cấp tính năng gợi ý mã, tự động hoàn thành, kiểm tra và sửa lỗi cú pháp, giúp tăng tốc độ lập trình và đảm bảo chất lượng mã nguồn.
- + Hỗ trợ mạnh mẽ từ cộng đồng: Với một cộng đồng đông đảo và nguồn tài liệu phong phú, Eclipse mang đến nhiều plugin hữu ích, dễ dàng mở rộng và tùy chỉnh theo nhu cầu sử dụng.

Draw.io

Draw.io là một công cụ thiết kế sơ đồ trực tuyến mạnh mẽ, được sử dụng phổ biến để tạo các sơ đồ UML, sơ đồ mạng, sơ đồ quy trình, và nhiều loại sơ đồ khác. Với giao diện thân thiện và dễ sử dụng, draw.io cho phép người dùng kéo thả các thành phần, tùy chỉnh hình dạng, và liên kết các đối tượng chỉ trong vài bước đơn giản. Công cụ này hoạt động trực tiếp trên trình duyệt, không yêu cầu cài đặt, và hỗ trợ lưu trữ trực tuyến qua Google Drive, OneDrive hoặc lưu trực tiếp trên máy tính.

Một trong những tính năng nổi bật của draw.io là thư viện biểu tượng đa dạng, từ các thành phần kỹ thuật đến các biểu tượng quản lý, giúp người dùng nhanh chóng xây dựng các sơ đồ chuyên nghiệp. Công cụ này còn hỗ trợ tích hợp với nhiều nền tảng như Atlassian Confluence và Jira, giúp các nhóm làm việc cộng tác hiệu quả hơn. Khả năng xuất file dưới nhiều định dạng như PNG, PDF, SVG hay XML giúp việc chia sẻ và chỉnh sửa sơ đồ trở nên dễ dàng hơn.

Với sự tiện lợi và linh hoạt, draw.io không chỉ phù hợp cho sinh viên học tập, mà còn là giải pháp lý tưởng cho các doanh nghiệp và tổ chức trong việc minh họa ý tưởng, phân tích hệ thống, hoặc lập kế hoạch dự án.

## PHẦN 2: NỘI DUNG

### CHƯƠNG 1: KHẢO SÁT HỆ THỐNG

#### 1.1 Tài liệu đặc tả yêu cầu người dùng

##### 1.1.1 Khảo sát sơ bộ

###### Mục tiêu:

Trong quá trình nghiên cứu và phát triển hệ thống quản lý tiệm tạp hóa HaUI, việc điều tra và khảo sát sơ bộ hệ thống hiện tại là bước đầu tiên và quan trọng. Mục tiêu của giai đoạn này là thu thập thông tin cần thiết để hiểu rõ về quy trình quản lý hiện tại, từ đó xác định được các yêu cầu và nhu cầu cụ thể cho hệ thống mới.

Qua quá trình khảo sát sơ bộ, nhóm nghiên cứu sẽ thu thập được dữ liệu đầu vào quan trọng, từ đó xác định được các yêu cầu chức năng và phi chức năng cho hệ thống quản lý tiệm tạp hóa. Bao gồm việc đánh giá các công cụ và công nghệ hiện có, cũng như xác định các cơ hội và thách thức trong quá trình phát triển hệ thống. Mục tiêu cuối cùng là xây dựng một hệ thống quản lý hiện đại, đáp ứng tốt nhất các nhu cầu của phần mềm quản lý tiệm tạp hóa, từ quản lý tài khoản, quản lý loại sản phẩm, quản lý sản phẩm, quản lý hóa đơn đến thống kê, xuất ra file PDF,... phù hợp với mục đích sử dụng.

###### Kết quả sơ bộ

- Hoạt động của hệ thống đang vận hành
  - Hàng tháng hệ thống căn cứ vào bảng chi tiết các hóa đơn của mỗi khách hàng để in ra báo cáo bán hàng: Doanh thu hàng tháng,...
- Điểm mạnh và điểm yếu của hệ thống đang vận hành
  - Điểm mạnh: Hệ thống hoạt động ổn định, đa dạng các loại sản phẩm, sản phẩm theo từng loại.
  - Điểm yếu: Hệ thống phải dùng nhiều nhân lực, có những giấy tờ thủ tục rườm rà, mất rất nhiều thời gian và công sức cho báo cáo và lên danh sách thu chi, đôi lúc còn bị sai sót, tính bảo mật chưa cao.

#### 1.1.2 Tài liệu đặc tả yêu cầu người dùng

- **Hoạt động của hệ thống**
  - Hệ thống cho phép quản trị viên đăng nhập để thêm, sửa, xóa thông tin về loại sản phẩm: Mã loại sản phẩm, Tên loại sản phẩm, Mô tả.
  - Hệ thống cho phép quản trị viên đăng nhập để thêm, sửa, xóa, tìm kiếm, sắp xếp thông tin về sản phẩm: Mã sản phẩm, Tên sản phẩm, Mã loại sản phẩm, Đơn giá, Số lượng.
  - Hệ thống cho phép quản trị viên đăng nhập để tạo hóa đơn bán hàng.

- Hệ thống cho phép quản trị viên thống kê doanh thu theo từng tháng.
- **Yêu cầu chức năng**
  - Đăng nhập: Hệ thống cho phép quản trị viên đăng nhập để thực hiện các chức năng của hệ thống.
  - Đăng ký: Hệ thống cho phép đăng ký tài khoản để truy cập sử dụng hệ thống.
  - Quản lý tài khoản: Xác minh quyền truy cập tài khoản khác.
  - Quản lý loại sản phẩm: Lưu trữ thông tin về loại sản phẩm.
  - Quản lý sản phẩm: Lưu trữ thông tin về sản phẩm.
  - Quản lý hóa đơn: Lưu trữ thông tin hóa đơn.
  - Thống kê: Thống kê doanh thu của tiệm theo từng tháng dựa vào những hóa đơn đã tạo.
- **Yêu cầu phi chức năng**
  - Hệ thống xử lý nhanh gọn, chính xác và thuận tiện.
  - Giao diện hài hòa, thân thiện và dễ sử dụng.
  - Quy trình phát triển phần mềm phù hợp để dễ dàng bảo trì và nâng cấp.
  - Có sự ràng buộc chặt chẽ về chất lượng, môi trường và chuẩn sử dụng.
  - Đảm bảo về mặt thời gian, bản quyền.
  - Chi phí chấp nhận được.

### 1.1.3 Mô hình hoá chức năng hệ thống

#### Biểu đồ use case

Các tác nhân của hệ thống: Theo phần đặc tả yêu cầu người dùng, có thể xác định được các actor như sau:

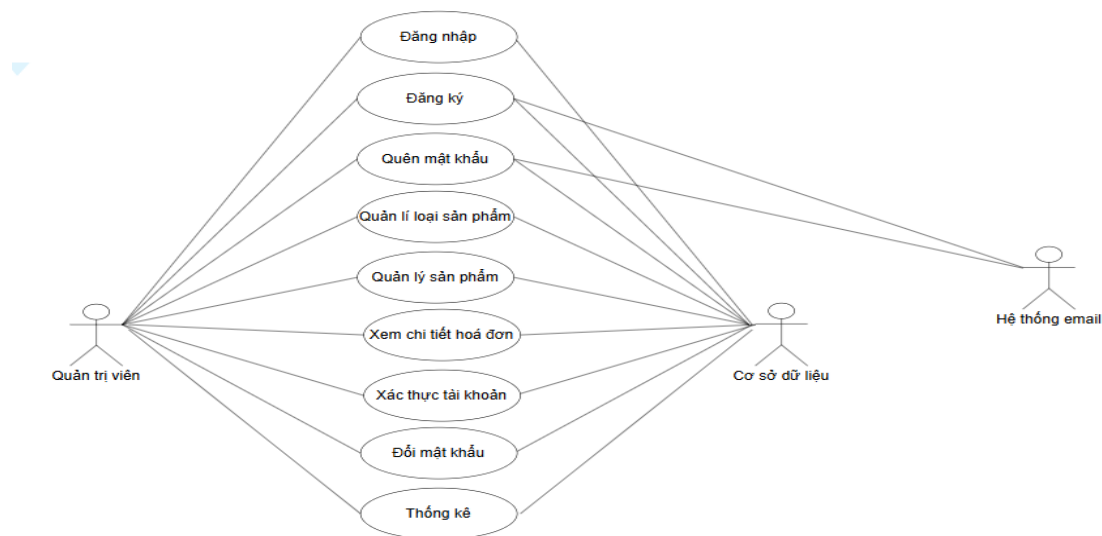
- Quản trị viên: Là tác nhân chính, đăng nhập để thực hiện các chức năng của hệ thống.
- Cơ sở dữ liệu: Tác nhân phụ, nơi lưu các dữ liệu để sử dụng.
- Hệ thống thư điện tử: Tác nhân phụ, sử dụng trong việc xác minh đăng ký tài khoản, hoặc lấy lại mật khẩu.

Các use case chính: Dựa theo việc xác định actor ở trên, ta xác định được các use case tương ứng:

- Quản lý loại sản phẩm
- Xem sản phẩm theo loại
- Xuất loại sản phẩm ra file PDF
- Quản lý sản phẩm
- Tìm kiếm sản phẩm
- Sắp xếp danh sách sản phẩm

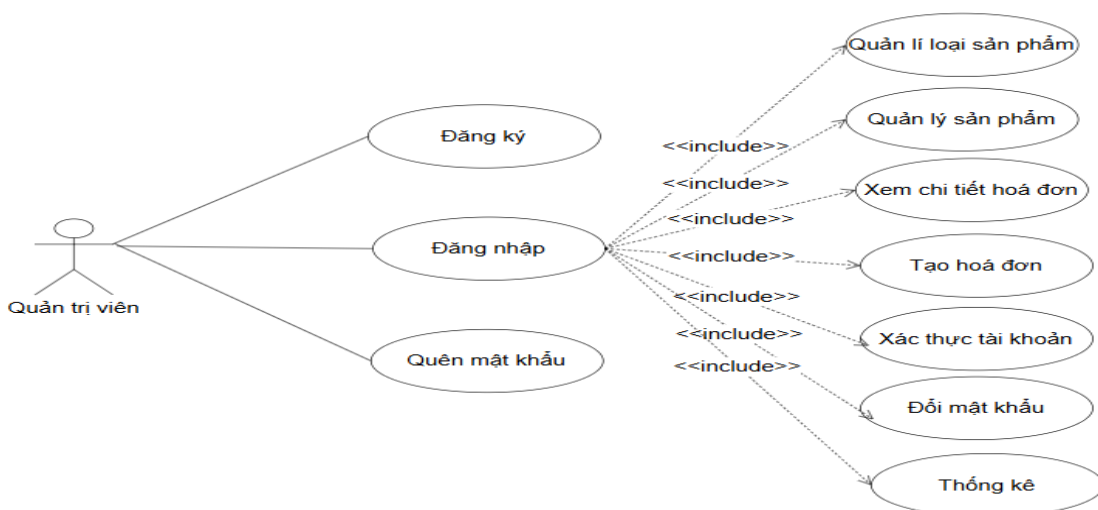
- Xuất sản phẩm ra file PDF
- Quản lý hóa đơn
- Xác thực tài khoản
- Đổi mật khẩu
- Thống kê
- Đăng nhập
- Đăng ký
- Tìm lại mật khẩu

### Biểu đồ use case tổng quát



Hình I: Biểu đồ use case tổng quát

### Biểu đồ phân rã use case



Hình II: Biểu đồ phân rã use case

## 1.1.4 Mô tả chi tiết các use case

### 1.1.4.1 Use case Đăng nhập (Nguyễn Ngọc Huy)

- **Mô tả:** Use case này cho phép người dùng truy cập vào hệ thống để sử dụng các chức năng quản lý sau khi đăng nhập thành công.
- **Luồng sự kiện:**
  - a. Luồng cơ bản:
    1. Use case bắt đầu khi người dùng khởi chạy phần mềm. Giao diện đăng nhập sẽ hiển thị lên màn hình.
    2. Người dùng nhập thông tin Tên đăng nhập và Mật khẩu, sau đó nhấn nút “Đăng nhập”. Hệ thống sẽ kiểm tra thông tin trong cơ sở dữ liệu. Nếu thông tin hợp lệ, hệ thống hiển thị thông báo “Đăng nhập thành công” và chuyển hướng đến trang chủ. Use case kết thúc.
  - b. Luồng rẽ nhánh:
    1. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.
    2. Tại bước 2 của luồng cơ bản, nếu người dùng để trống bất kỳ trường thông tin nào, hệ thống sẽ thông báo lỗi “Vui lòng nhập đầy đủ thông tin” và quay lại bước 1.
    3. Tại bước 2, nếu Tên đăng nhập hoặc Mật khẩu không chính xác (không tồn tại trong cơ sở dữ liệu), hệ thống sẽ hiển thị thông báo lỗi “Tên đăng nhập hoặc mật khẩu không chính xác.” và quay lại bước 1.
    4. Tại bước 2, nếu tài khoản chưa được cấp quyền truy cập, hệ thống sẽ hiển thị thông báo lỗi “Tài khoản của bạn không có quyền truy cập. Vui lòng liên hệ Admin.”.
    5. Trước bước 2, nếu người dùng chọn “Đăng ký”, hệ thống sẽ chuyển đến use case “Đăng ký”.
    6. Trước bước 2, nếu người dùng chọn “Lấy lại mật khẩu”, hệ thống sẽ chuyển đến use case “Lấy lại mật khẩu”.
- **Các yêu cầu đặc biệt:**

Không có.
- **Tiền điều kiện:**

Không có.

- **Hậu điều kiện:**

Nếu đăng nhập thành công, người dùng sẽ được chuyển đến hệ thống. Nếu không thành công, giao diện đăng nhập vẫn được giữ nguyên.

- **Điểm mở rộng:**

Không có.

#### 1.1.4.2 Use case Đăng kí (Nguyễn Ngọc Huy)

- **Mô tả:** Use case này cho phép người dùng tạo tài khoản mới để truy cập vào hệ thống.

- **Luồng sự kiện:**

a. Luồng cơ bản

1. Use case bắt đầu khi người dùng chọn “Đăng ký” từ giao diện đăng nhập. Giao diện đăng ký sẽ được hiển thị lên màn hình.
2. Người dùng điền các thông tin gồm: Tên đăng nhập, Mật khẩu, Nhập lại mật khẩu, Email, sau đó nhấn “Gửi mã”. Hệ thống sẽ gửi mã xác minh gồm 6 chữ số tới email đã nhập và hiển thị thông báo “Mã xác nhận đã được gửi vào email. Vui lòng kiểm tra.”
3. Người dùng nhập mã xác minh nhận được qua email và nhấn “Đăng ký”. Nếu thông tin hợp lệ, hệ thống sẽ tạo tài khoản mới, lưu vào cơ sở dữ liệu và hiển thị thông báo “Đăng ký thành công”. Use case kết thúc.

b. Luồng rẽ nhánh

1. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.
2. Tại bước 2, nếu email không đúng định dạng hoặc không tồn tại, hệ thống sẽ báo lỗi “Có lỗi trong quá trình lấy mã xác nhận. Vui lòng thử lại.” và quay lại bước 1.
3. Tại bước 3, nếu người dùng để trống bất kỳ trường thông tin nào, hệ thống sẽ báo lỗi “Vui lòng nhập đầy đủ thông tin” và quay lại trước bước 3.
4. Tại bước 3, nếu mã xác minh nhập sai, hệ thống sẽ hiển thị thông báo “Vui lòng kiểm tra lại mã xác nhận” và quay lại trước bước 3.
5. Tại bước 3, nếu tên đăng nhập hoặc email đã tồn tại trong cơ sở dữ liệu, hệ thống sẽ hiển thị thông báo “Đã tồn tại tài khoản với tên tài khoản hoặc email này. Vui lòng thử lại.” và quay lại trước bước 3.



6. Trước bước 2, nếu người dùng chọn “Đăng nhập”, hệ thống sẽ chuyển hướng sang use case “Đăng nhập”.

- Các yêu cầu đặc biệt:  
Không có.
- Tiên điều kiện:  
Không có.
- Hậu điều kiện:  
Nếu đăng ký thành công, hệ thống sẽ tạo tài khoản mới, mã hóa mật khẩu và lưu thông tin vào cơ sở dữ liệu.
- Điểm mở rộng:  
Không có.

#### **1.1.4.3 Use case Tạo hoá đơn (Lê Thị Thanh Thảo)**

- Mô tả: Use case này cho phép người dùng tạo hóa đơn bán hàng.
- Luồng sự kiện
  - a. Luồng cơ bản:

1. Use case này bắt đầu khi người dùng nhấn chọn "Tạo hóa đơn" từ màn hình trang chủ. Hệ thống sẽ hiển thị hộp thoại cho phép người dùng nhập vào các thông tin để tạo hóa đơn.

2. Người dùng nhập thông tin khách hàng và nhấn chọn “Thêm sản phẩm”. Hệ thống sẽ truy xuất cơ sở dữ liệu và hiển thị lên màn hình bảng chọn sản phẩm.

3. Người dùng nhấn đúp để chọn sản phẩm và nhấn “Ok”. Hệ thống sẽ hiển thị hộp thoại yêu cầu người dùng nhập số lượng.

4. Người dùng nhập số lượng sản phẩm tương ứng. Hệ thống sẽ hiển thị sản phẩm cùng số lượng lên bảng danh sách sản phẩm.

5. Người dùng nhấn chọn “Tạo”. Hệ thống sẽ lưu vào cơ sở dữ liệu hóa đơn vừa tạo và hiển thị một thông báo “Tạo hóa đơn thành công”. Use case kết thúc.

- b. Luồng rẽ nhánh:

1. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.

2. Tại bước 3 của luồng cơ bản, nếu số lượng sản phẩm không đủ, hệ thống sẽ hiển thị một thông báo lỗi “Không đủ số lượng”. Use case quay lại bước 2.

3. Tại bước 5 của luồng cơ bản, nếu bất kì trường nào bị bỏ trống, hệ thống sẽ hiển thị lên một thông báo lỗi “Vui lòng nhập đầy đủ thông tin”. Use case quay lại trước bước 1.

- **Các yêu cầu đặc biệt**

Không có.

- **Tiền điều kiện**

Người dùng phải đăng nhập vào hệ thống.

- **Hậu điều kiện**

Hệ thống phải cập nhật các thao tác thêm hóa đơn của người dùng vào cơ sở dữ liệu.

- **Điểm mở rộng**

Không có.

#### **1.1.4.4 Use case Xác thực tài khoản (Lê Thị Thanh Thảo)**

- **Mô tả:** Use case này cho phép quản trị viên xác thực cho những tài khoản khác.

- **Luồng sự kiện**

a. Luồng cơ bản.

1. Use case này bắt đầu khi người dùng nhấn chọn “Xác minh tài khoản” từ giao diện đăng nhập. Hệ thống sẽ hiển thị lên màn hình giao diện xác minh tài khoản.
2. Người dùng nhấn đúp vào tài khoản muốn xác minh, hệ thống sẽ hiển thị lên hộp thoại yêu cầu xác minh.
3. Người dùng nhấn chọn “Yes”, hệ thống sẽ xác minh cho tài khoản và lưu vào cơ sở dữ liệu. Use case kết thúc.

b. Luồng rẽ nhánh

1. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.
2. Tại bước 3 của luồng cơ bản, nếu người dùng nhấn chọn “No”, hộp thoại sẽ biến mất. Use case quay lại trước bước 2.

- **Các yêu cầu đặc biệt**

Không có.

- **Tiền điều kiện**

Người dùng phải đăng nhập vào hệ thống.

- Hậu điều kiện

Nếu xác thực thành công, hệ thống sẽ cập nhật trạng thái tài khoản và lưu vào cơ sở dữ liệu.

- Điểm mở rộng

Không có.

#### 1.1.4.5 Use case Quản lý sản phẩm (Nguyễn Đình Thường)

- **Mô tả:** Use case này cho phép người dùng quản lý (thêm, sửa, xóa, tìm kiếm, sắp xếp, xuất file PDF) sản phẩm.
- **Luồng sự kiện**
  - a. Luồng cơ bản
    1. Use case này bắt đầu khi người dùng nhấn chọn “Quản lý loại sản phẩm” từ màn hình trang chủ. Hệ thống sẽ truy xuất cơ sở dữ liệu để lấy ra thông tin về các loại sản phẩm và hiển thị giao diện quản lý loại sản phẩm lên màn hình.
    2. Thêm loại sản phẩm
      - i. Người dùng chọn nút “Thêm” từ giao diện quản lý loại sản phẩm. Hệ thống sẽ hiển thị lên một hộp thoại cho phép người dùng thêm loại sản phẩm.
      - ii. Người dùng nhập các thông tin bao gồm: Mã loại sản phẩm, Tên loại sản phẩm, Mô tả và nhấn chọn “Lưu”. Hệ thống sẽ tạo mới loại sản phẩm và lưu vào cơ sở dữ liệu đồng thời hiển thị lên một thông báo “Thêm thành công”. Use case kết thúc.
    3. Sửa sản phẩm
      - i. Người dùng chọn một sản phẩm từ giao diện quản lý sản phẩm và nhấn chọn “Sửa”. Hệ thống sẽ hiển thị lên một hộp thoại cho phép người dùng sửa loại sản phẩm đó.
      - ii. Người dùng sửa các thông tin bao gồm: Tên sản phẩm, loại sản phẩm, số lượng, giá và nhấn chọn “Lưu”. Hệ thống sẽ truy xuất vào cơ sở dữ liệu để tìm loại sản phẩm tương ứng và lưu giá trị mới thay thế đồng thời hiển thị lên màn hình một thông báo “Sửa thành công”. Use case kết thúc.
    4. Xóa sản phẩm
      - i. Người dùng chọn một sản phẩm từ giao diện quản lý sản phẩm và nhấn chọn “Xóa”. Hệ thống sẽ hiển thị lên một hộp thoại xác nhận.
      - ii. Người dùng nhấn chọn “Yes”, hệ thống sẽ xóa sản phẩm được chọn ra khỏi cơ sở dữ liệu. Use case kết thúc.
    5. Tìm kiếm loại sản phẩm

- i. Trong giao diện quản lý sản phẩm, người dùng nhập từ khóa vào ô tìm kiếm, hệ thống sẽ bắt đầu tìm kiếm trong cơ sở dữ liệu sau 0,5 giây kể từ khi người dùng ngừng nhập và hiển thị lên màn hình kết quả tương ứng. Use case kết thúc.
6. Sắp xếp sản phẩm theo giá sản phẩm
  - i. Người dùng cách sắp xếp sản phẩm từ combobox . Hệ thống sẽ sắp xếp danh sách sản phẩm và hiển thị lên màn hình. Use case kết thúc.
7. Xuất danh sách loại sản phẩm
  - i. Người dùng nhấn chọn “Xuất ra PDF” trong giao diện quản lý sản phẩm, hệ thống sẽ hiển thị lên một hộp thoại tùy chọn.
  - ii. Người dùng chọn những tùy chọn phù hợp trong hộp thoại và nhấn chọn “Print”, hệ thống sẽ in tùy vào những lựa chọn của người dùng và hiển thị lên một thông báo thành công. Use case kết thúc.
- b. Luồng rẽ nhánh
  1. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.
  2. Tại bước 2.2 của luồng cơ bản, nếu người dùng thêm một loại sản phẩm với mã sản phẩm đã tồn tại, hệ thống sẽ hiển thị một thông báo lỗi “Thêm thất bại”. Use case quay lại bước 2.1.
  3. Tại các bước 2.2 và 3.2 của luồng cơ bản, nếu người dùng nhập thiếu thông tin tại các bắt kì trường nào thì hệ thống sẽ hiển thị một thông báo lỗi “Vui lòng nhập đầy đủ thông tin”. Use case quay lại bước 2.1 hoặc 3.1 tương ứng.
  4. Tại các bước 3.1 và 4.1 của luồng cơ bản, nếu người dùng không chọn sản phẩm, hệ thống sẽ hiển thị một thông báo lỗi “Vui lòng chọn loại sản phẩm”. Use case quay lại bước 1.
- **Các yêu cầu đặc biệt**  
Không có.
- **Tiền điều kiện**  
Người dùng phải đăng nhập vào hệ thống.
- **Hậu điều kiện**  
Hệ thống phải cập nhật các thao tác thêm, sửa xoá sản phẩm của người dùng vào cơ sở dữ liệu.
- **Điểm mở rộng**  
Không có.

#### 1.1.4.5 Use case Xem chi tiết hoá đơn(Nguyễn Đình Thường)

- **Mô tả:** Use case này cho phép người dùng quản lý (tìm kiếm theo mã/ngày tạo, xem chi tiết) hóa đơn.
- **Luồng sự kiện**
  - a. Luồng cơ bản
    1. Use case này bắt đầu khi người dùng nhấn chọn "Quản lý hóa đơn" từ màn hình trang chủ.
    2. Hệ thống sẽ truy xuất cơ sở dữ liệu để lấy ra thông tin về các hóa đơn và hiển thị giao diện quản lý hóa đơn lên màn hình.
    3. Tìm kiếm hóa đơn
      - i. Trong giao diện quản lý hóa đơn, người dùng chọn loại tìm kiếm (theo mã hoặc theo ngày tạo) từ combobox.
      - ii. Người dùng nhập từ khóa (mã hóa đơn hoặc khoảng thời gian) vào ô tìm kiếm.
      - iii. Hệ thống sẽ bắt đầu tìm kiếm trong cơ sở dữ liệu sau khi người dùng ngưng nhập 0.5s và hiển thị lên màn hình kết quả tương ứng. Use case kết thúc.
    4. Xem chi tiết hóa đơn
      - i. Người dùng chọn một hóa đơn từ danh sách kết quả tìm kiếm hoặc từ giao diện quản lý hóa đơn.
      - ii. Hệ thống sẽ hiển thị chi tiết hóa đơn bao gồm: Mã hóa đơn, Tên khách hàng, Sản phẩm, Số lượng, Tổng tiền, Ngày tạo, và các thông tin liên quan khác (nếu có). Use case kết thúc.
  - b. Luồng rẽ nhánh
    1. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case nếu không kết nối được cơ sở dữ liệu thì hệ thống sẽ hiển thị một thông báo lỗi và use case kết thúc.
    2. Tại bước 2.2 của luồng cơ bản, nếu người dùng thêm một hóa đơn với mã hóa đơn đã tồn tại, hệ thống sẽ hiển thị một thông báo lỗi "Thêm thất bại". Use case quay lại bước 2.1.
    3. Tại bước 2.2 của luồng cơ bản, nếu người dùng nhập thiếu thông tin tại các bất kì trường nào thì hệ thống sẽ hiển thị một thông báo lỗi "Vui lòng nhập đầy đủ thông tin". Use case quay lại bước 2.1.
    4. Tại bước 3.2 của luồng cơ bản, nếu không tìm thấy hóa đơn nào phù hợp với từ khóa tìm kiếm, hệ thống sẽ hiển thị một thông báo "Không tìm thấy hóa đơn". Use case kết thúc.
- **Các yêu cầu đặc biệt**

Không có.
- **Tiền điều kiện**

Người dùng phải đăng nhập vào hệ thống.

- **Hậu điều kiện**

Không có.

- **Điểm mở rộng**

Có thể mở rộng use case để bao gồm các chức năng khác như sửa, xóa, lọc hóa đơn, v.v.

### 1.1.5 Mô hình hoá dữ liệu của hệ thống

#### Xác định các lớp và thuộc tính

- Mỗi quản trị viên bao gồm các thuộc tính: Mã quản trị viên, Tên đăng nhập, Email, Mật khẩu, Đã xác minh. Sau khi đăng nhập hợp lệ, quản trị viên có thể thực hiện các tác vụ quản lý tài khoản, quản lý loại sản phẩm, quản lý sản phẩm, quản lý hóa đơn, thống kê,...
- Thông tin về loại sản phẩm bao gồm: Mã loại sản phẩm, Tên loại sản phẩm, Mô tả.
- Thông tin về sản phẩm bao gồm: Mã sản phẩm, Tên sản phẩm, Mã loại sản phẩm, Đơn giá, Số lượng.
- Quản trị viên có thể tạo hóa đơn bán hàng. Thông tin về hóa đơn bao gồm: Mã hóa đơn, Tên khách hàng, Mã quản trị viên, Ngày tạo, Danh sách sản phẩm và số lượng, Tổng tiền.

Do đó, ta xác định được 4 lớp cơ bản:

*Bảng I: Các lớp cơ bản*

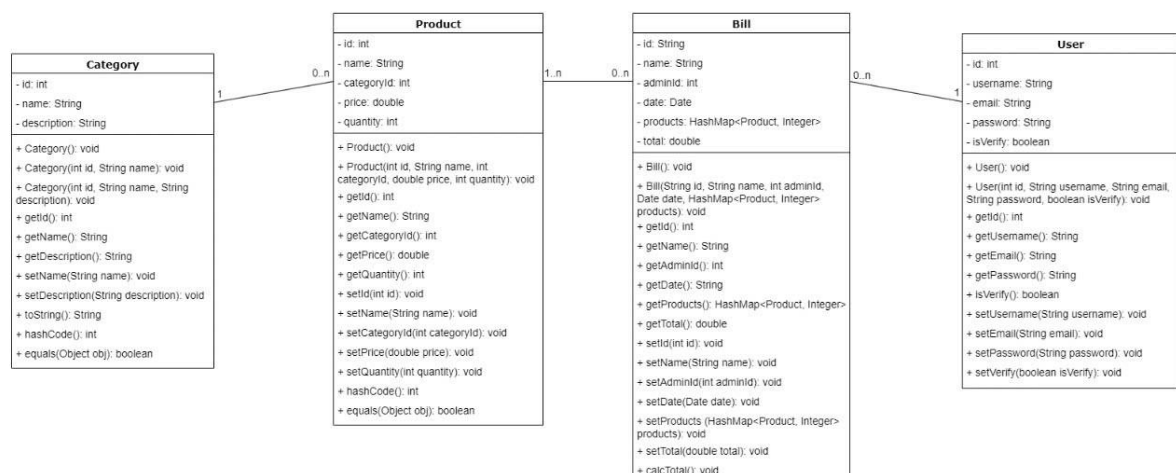
<b>Lớp Quản trị viên (User)</b>	<b>Lớp Loại sản phẩm (Category)</b>
<ul style="list-style-type: none"><li>• Mã quản trị viên (id: int)</li><li>• Tên đăng nhập (username: String)</li><li>• Email (email: String)</li><li>• Mật khẩu (password: String)</li><li>• Đã xác minh (isVerify: boolean)</li></ul>	<ul style="list-style-type: none"><li>• Mã loại sản phẩm (id: int)</li><li>• Tên loại sản phẩm (name: String)</li><li>• Mô tả (description: String)</li></ul>

<b>Lớp Sản phẩm (Product)</b> <ul style="list-style-type: none"> <li>Mã sản phẩm (id: int)</li> <li>Mã sản phẩm (name: String)</li> <li>Mã loại sản phẩm (categoryId: int)</li> <li>Đơn giá (price: double)</li> <li>Số lượng (quantity: int)</li> </ul>	<b>Lớp Hoá đơn (Bill)</b> <ul style="list-style-type: none"> <li>Mã hóa đơn (id: String)</li> <li>Tên khách hàng (name: String)</li> <li>Mã quản trị viên (adminId: int)</li> <li>Ngày tạo (date: Date)</li> <li>Danh sách sản phẩm (products: HashMap)</li> <li>Tổng tiền (total: double)</li> </ul>
--	---

### Quy tắc nghiệp vụ giữa các lớp

- Mỗi loại sản phẩm có thể có 0, 1 hoặc nhiều sản phẩm. Mỗi sản phẩm chỉ thuộc về 1 và chỉ 1 loại sản phẩm.
- Mỗi hóa đơn có thể có 1 hoặc nhiều sản phẩm. Mỗi sản phẩm có thể thuộc về 0, 1 hoặc nhiều hóa đơn.
- Mỗi quản trị viên có thể tạo 0, 1 hoặc nhiều hóa đơn. Mỗi hóa đơn chỉ do 1 và chỉ 1 quản trị viên tạo

### Quan hệ giữa các lớp



Hình III: Biểu diễn mối quan hệ giữa các lớp

## CHƯƠNG 2: THIẾT KẾ HỆ THỐNG

### 2.1 Thiết kế giao diện

#### 2.1.1 Giao diện đăng nhập

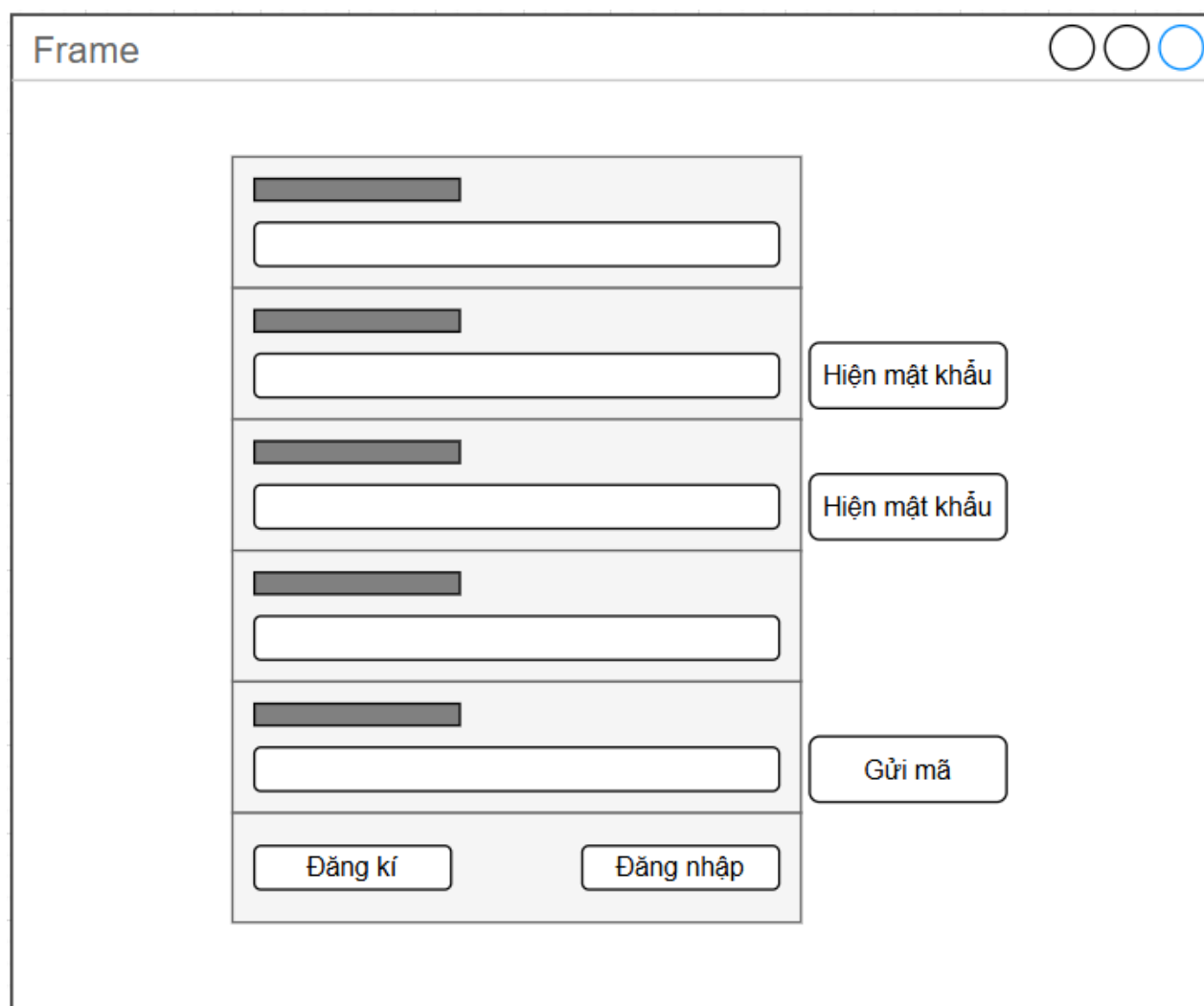
The image shows a UI design for a login interface. It is contained within a window titled "Frame" with standard window controls (minimize, maximize, close) in the top right corner. The interface features a central login form with the following elements:

- A square icon with a diagonal cross (X) at the top center.
- A horizontal bar (likely a logo or separator) above the first input field.
- A text input field for the username.
- A horizontal bar (likely a logo or separator) above the second input field.
- A text input field for the password.
- A button labeled "Hiện mật khẩu" (Show password) to the right of the password input field.
- Two buttons at the bottom: "Đăng nhập" (Login) and "Đăng kí" (Register).
- A button labeled "Lấy lại mật khẩu" (Reset password) at the bottom center.

Hình IV: Thiết kế giao diện đăng nhập



### 2.1.2 Giao diện đăng kí



Frame

Form structure:

- Row 1: Grey label bar, Input field
- Row 2: Grey label bar, Input field, Button: Hiện mật khẩu
- Row 3: Grey label bar, Input field, Button: Hiện mật khẩu
- Row 4: Grey label bar, Input field
- Row 5: Grey label bar, Input field, Button: Gửi mã
- Bottom: Button: Đăng kí, Button: Đăng nhập

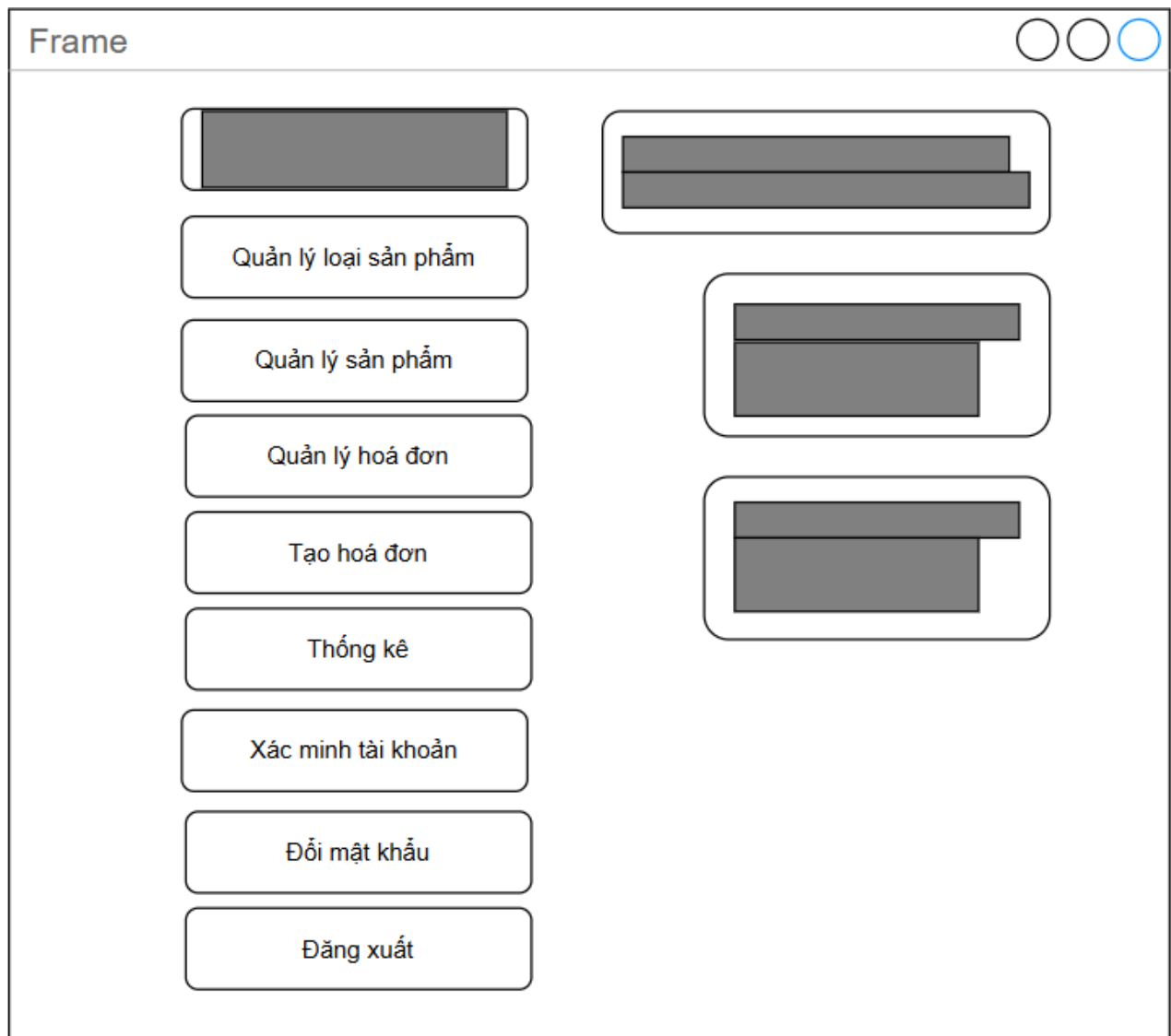
Hình V: Thiết kế giao diện đăng ký

### 2.1.3 Giao diện quên mật khẩu

The image shows a UI design for a password reset interface. It is contained within a window titled "Frame" with standard macOS-style window controls (red, yellow, and blue buttons) in the top right corner. At the top center of the window is a square icon with an 'X' inside. Below this icon is a light gray rectangular panel. Inside this panel, there are two input fields, each preceded by a dark gray rectangular label. To the right of the second input field is a button labeled "Hiện mật khẩu". Below the input fields, there are two buttons: "Đăng nhập" and "Đăng kí". At the bottom of the panel is a button labeled "Lấy lại mật khẩu".

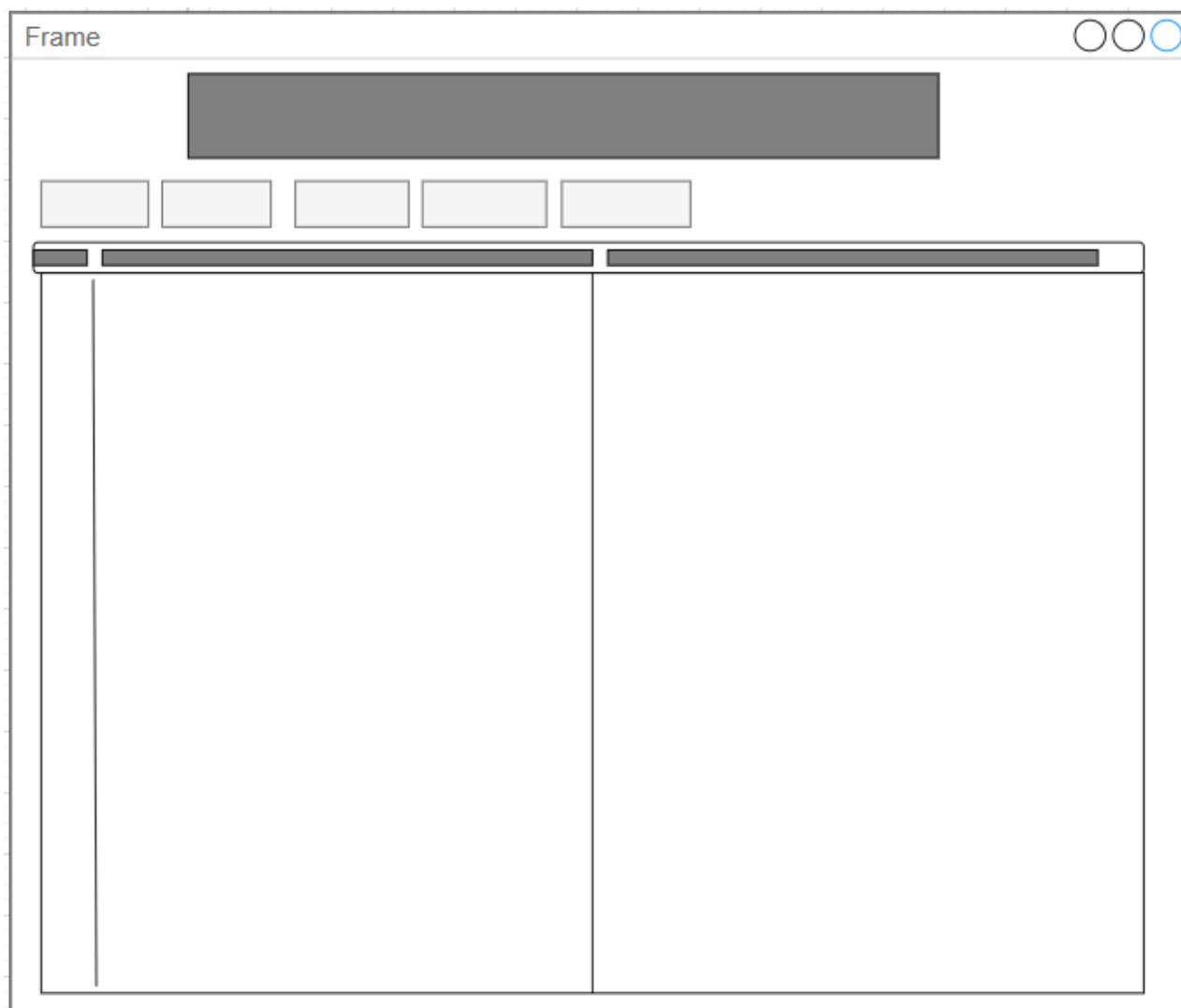
Hình VI: Thiết kế giao diện quên mật khẩu

### 2.1.4 Giao diện màn hình chính



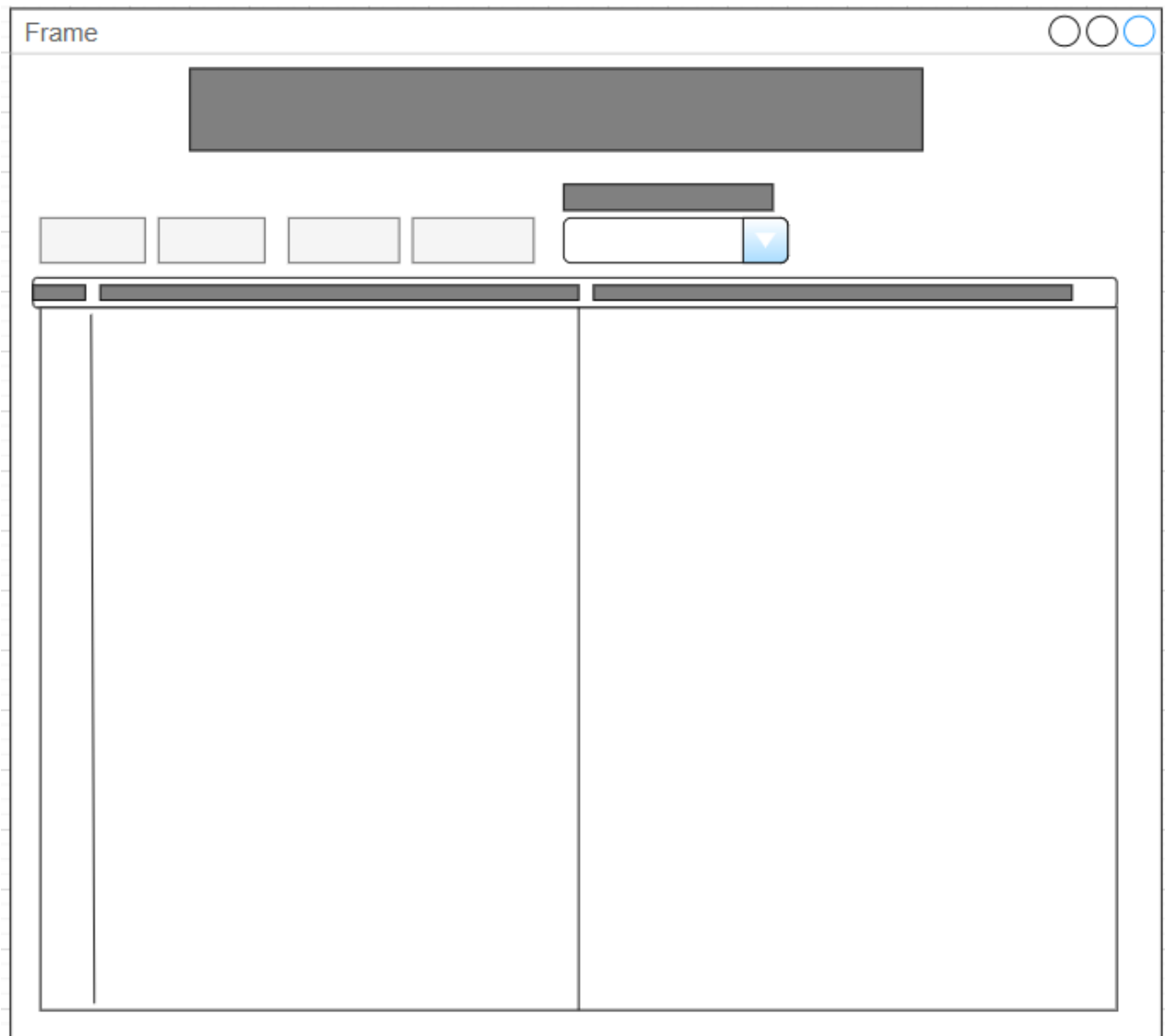
*Hình VII: Thiết kế giao diện màn hình chính*

### 2.1.5 Giao diện quản lý loại sản phẩm



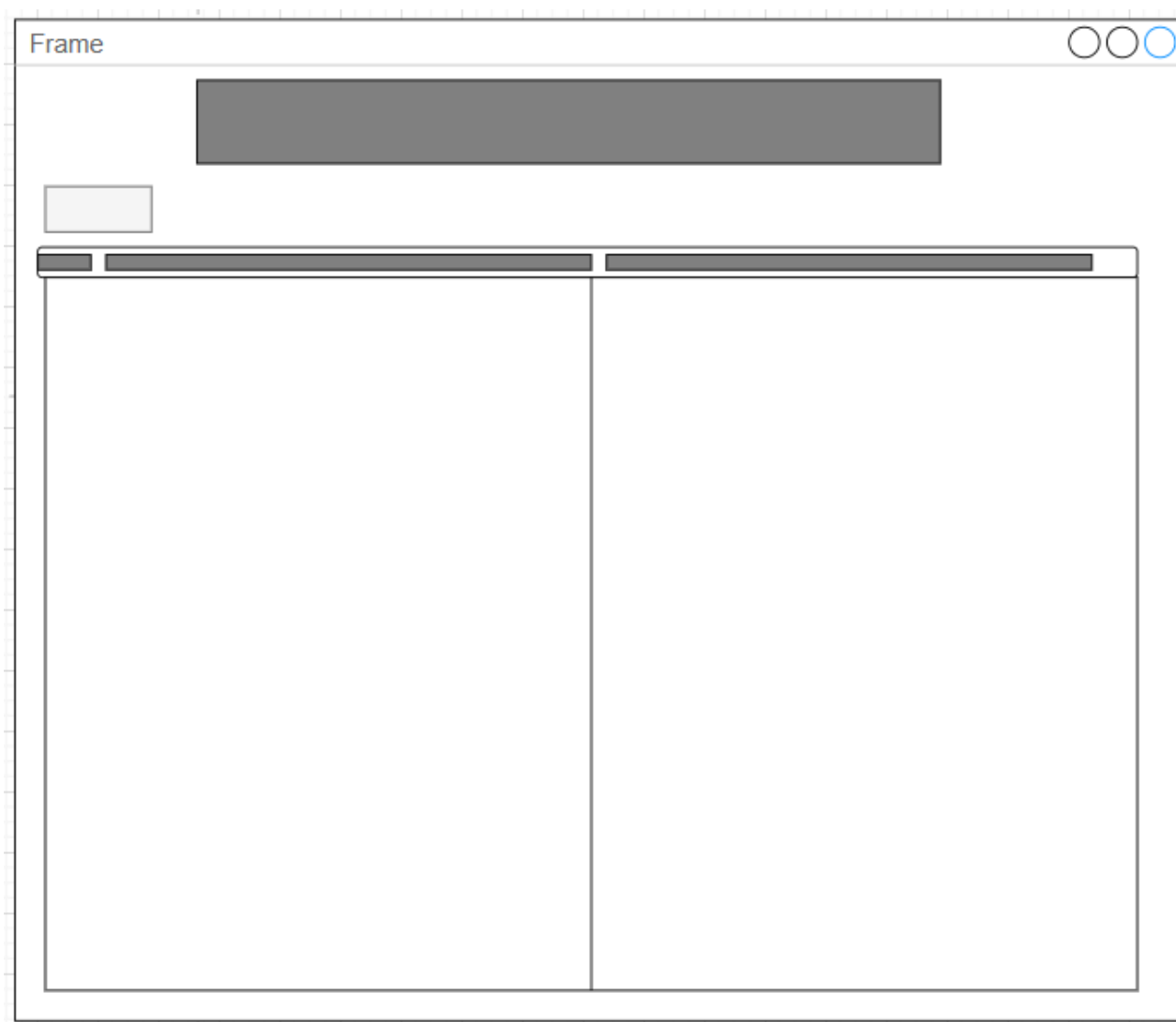
*Hình VIII: Thiết kế giao diện quản lý loại sản phẩm*

### 2.1.6 Giao diện quản lý sản phẩm



*Hình IX: Thiết kế giao diện quản lý sản phẩm*

### 2.1.7 Giao diện quản lý hoá đơn



*Hình X: Thiết kế giao diện quản lý hoá đơn*

### 2.2 Thiết kế dữ liệu - Ánh xạ lớp sang bảng

#### Bảng User (Quản trị viên)

Thuộc tính	Tên thuộc tính	Kiểu dữ liệu
id	Mã quản trị viên	Integer
username	Tên đăng nhập	String
email	Email	String
password	Mật khẩu	String
isVerify	Đã xác minh	Boolean

*Bảng II: Bảng User (Quản trị viên)*

**Bảng Category** (*Loại sản phẩm*)

Thuộc tính	Tên thuộc tính	Kiểu dữ liệu
id	Mã loại sản phẩm	Integer
name	Tên loại sản phẩm	String
description	Mô tả	String

*Bảng III: Bảng Category (Loại sản phẩm)***Bảng Product** (*Sản phẩm*)

Thuộc tính	Tên thuộc tính	Kiểu dữ liệu
id	Mã sản phẩm	Integer
name	Tên sản phẩm	String
categoryId	Mã loại sản phẩm	Integer
price	Đơn giá	Double
quantity	Số lượng	Integer

*Bảng IV: Bảng Product (Sản phẩm)***Bảng Bill** (*Hoá đơn*)

Thuộc tính	Tên thuộc tính	Kiểu dữ liệu
id	Mã hóa đơn	String
name	Tên khách hàng	String
adminId	Mã quản trị viên	Integer
date	Ngày tạo	Date
products	Danh sách sản phẩm	HashMap
total	Tổng tiền	Double

*Bảng V: Bảng Bill (Hoá đơn)*

## CHƯƠNG 3: CÀI ĐẶT VÀ TRIỂN KHAI

### 3.1 Giới thiệu công cụ

Phần mềm “Quản lý tiệm tạp hóa Xanh” được cài đặt và triển khai dựa trên công cụ Eclipse và sử dụng file nhị phân để lưu trữ dữ liệu, dự án được chạy trên JDK 17.

### 3.2 Giới thiệu về Eclipse

Eclipse là một môi trường phát triển tích hợp (IDE) phổ biến được sử dụng chủ yếu để phát triển các ứng dụng bằng Java.

Eclipse được phát triển ban đầu bởi IBM vào năm 2001 và sau đó được trao cho Quỹ Eclipse (Eclipse Foundation), một tổ chức phi lợi nhuận quản lý và phát triển nền tảng này. Kể từ đó, Eclipse đã trở thành một trong những IDE được sử dụng rộng rãi nhất trên thế giới.

Các tính năng chính:

- Hỗ trợ mạnh mẽ cho việc phát triển ứng dụng Java.
- Hỗ trợ nhiều plugin từ Eclipse Marketplace để mở rộng các chức năng.
- Cung cấp công cụ biên dịch, gỡ lỗi và refactoring.
- Hỗ trợ làm việc nhóm với các hệ thống quản lý phiên bản như Git, SVN,...

Cài đặt và sử dụng:

- Tải Eclipse IDE từ trang web chính thức của Eclipse. (Eclipse: <https://www.eclipse.org/>)
- Cài đặt dự án đã có sẵn bằng cách chọn File > Import > Existing Projects into Workspace và chọn đường dẫn tới dự án.



## CHƯƠNG 4: THỰC HIỆN BÀI TOÁN

Bài toán sẽ được cài đặt theo mô hình MVC (Model – View – Controller).

- Model: chứa các lớp đại diện cho dữ liệu, đối tượng làm việc:
  - User: Đại diện cho quản trị viên.
  - Category: Đại diện cho loại sản phẩm.
  - Product: Đại diện cho sản phẩm.
  - Bill: Đại diện cho hóa đơn.
- View: cung cấp giao diện người dùng, hiển thị dữ liệu và nhận đầu vào từ người dùng. Trong bài toán này giao diện người dùng được cài đặt bằng Java Swing.
- Controller: là các lớp trung gian để xử lý dữ liệu, thao tác với Model và đưa dữ liệu lên View. Các lớp này quản lý luồng dữ liệu giữa Model và View.

### 4.1 Các tiện ích (Lớp util)

#### 4.1.1 Làm việc với form

Hệ thống thực hiện kiểm tra dữ liệu được nhập vào đã đủ chưa thông qua hàm `validateForm()` và đưa form về mặc định thông qua `resetForm()`

```
package util;

import java.awt.Component;

public class FormUtils {
    public static boolean ValidateForm(JPanel inputPanel) {
        List<JTextField> jtfs = new ArrayList<>();

        collectTextFields(inputPanel, jtfs);

        return jtfs.stream().noneMatch(jtf -> jtf.getText().isEmpty());
    }

    public static void resetForm(JPanel inputPanel) {
        List<JTextField> jtfs = new ArrayList<>();

        collectTextFields(inputPanel, jtfs);

        jtfs.forEach(jtf -> jtf.setText(""));
    }
}
```

```

private static void collectTextFields(JPanel panel, List<JTextField> jtfs) {
    Component[] components = panel.getComponents();
    for (Component component : components) {
        if (component instanceof JTextField) {
            jtfs.add((JTextField) component);
        } else if (component instanceof JPanel) {
            collectTextFields((JPanel) component, jtfs);
        }
    }
}
}
}
}

```

#### 4.1.2 Thao tác đọc ghi file

Hệ thống thực hiện thao tác với file thông qua hàm `writeToFile()`, `readFromFile()` và `appendObject()` giúp tạo luồng nhập/xuất dữ liệu vào/ra file. Các đối tượng sẽ được lưu tại các file .bin tương ứng.

```

package util;

import java.io.File;

public class FileConnector<T> {
    public void writeToFile(String filePath, List<T> data) throws IOException {
        File fileName = new File(new File("").getAbsolutePath().concat(filePath));
        try (FileOutputStream fos = new FileOutputStream(fileName);
             ObjectOutputStream oos = new ObjectOutputStream(fos)) {
            oos.writeObject(data);
        }
    }

    @SuppressWarnings("unchecked")
    public List<T> readFromFile(String filePath) throws IOException, ClassNotFoundException {
        File file = new File(new File("").getAbsolutePath().concat(filePath));
        if (!file.exists() || file.length() == 0) {
            return new ArrayList<>();
        }
        try (FileInputStream fis = new FileInputStream(file); ObjectInputStream ois = new ObjectInputStream(fis)) {
            return (List<T>) ois.readObject();
        }
    }

    public void appendObject(String fileName, T newObject) throws IOException, ClassNotFoundException {
        List<T> objects = readFromFile(fileName);
        objects.add(newObject);
        writeToFile(fileName, objects);
    }
}

```

Các đối tượng thao tác với dữ liệu lưu trong file thông qua một interface – DAO.

Lớp interface `DAO<T>` (Lớp DAO trả về một kiểu T)

```

package dao;

import java.io.IOException;

public interface DAO<T> {
    T get(Predicate<T> predicate) throws ClassNotFoundException, IOException;

    List<T> getAll() throws ClassNotFoundException, IOException;

    boolean add(T t) throws ClassNotFoundException, IOException;

    boolean update(T t) throws ClassNotFoundException, IOException;

    boolean delete(T t) throws ClassNotFoundException, IOException;
}

```

Triển khai các đối tượng sử dụng DAO. Ví dụ cho lớp BillDAO:

```

package dao;

import java.io.IOException;

public class BillDAO implements DAO<Bill> {
    private final String FILE_PATH = "/src/db/bills.bin";
    private final FileConnector<Bill> fileConnector = new FileConnector<Bill>();

    @Override
    public Bill get(Predicate<Bill> predicate) throws ClassNotFoundException, IOException {
        List<Bill> bills = fileConnector.readFromFile(FILE_PATH);
        for (Bill bill : bills) {
            if (predicate.test(bill)) {
                return bill;
            }
        }
        return null;
    }

    @Override
    public List<Bill> getAll() throws ClassNotFoundException, IOException {
        return fileConnector.readFromFile(FILE_PATH);
    }

    @Override
    public boolean add(Bill bill) throws ClassNotFoundException, IOException {
        fileConnector.appendObject(FILE_PATH, bill);
        return true;
    }
}

```

```

@Override
public boolean update(Bill updatedBill) throws ClassNotFoundException, IOException {
    List<Bill> bills = fileConnector.readFromFile(FILE_PATH);
    for (int i = 0; i < bills.size(); i++) {
        if (bills.get(i).getId() == updatedBill.getId()) {
            bills.set(i, updatedBill);
            break;
        }
    }
    fileConnector.writeToFile(FILE_PATH, bills);
    return true;
}

@Override
public boolean delete(Bill deletedBill) throws ClassNotFoundException, IOException {
    List<Bill> bills = fileConnector.readFromFile(FILE_PATH);
    bills.removeIf(user -> user.equals(deletedBill));
    fileConnector.writeToFile(FILE_PATH, bills);
    return true;
}
}

```

Các lớp khác sử dụng DAO khác được cài đặt tương tự với BillDAO.

### 4.1.3 Gửi Email

Hệ thống thực hiện gửi email theo yêu cầu của người dùng trong các chức năng đăng ký, lấy lại mật khẩu. Chương trình sử dụng thư viện javax.mail để cài đặt chức năng gửi email.

Hệ thống sẽ tùy thuộc vào yêu cầu của người dùng để gửi các thông tin cần thiết như mã dùng một lần, hoặc mật khẩu mới.

```

package util;

import java.util.Properties;

public class SendMail {
    private String otp;
    private final String EMAIL = " ";
    private final String PASSWORD = " ";
    private final String HOST_NAME = "smtp.gmail.com";
    private final int SSL_PORT = 465;

    public String getOtp() {
        return otp;
    }

    public void setOtp(String otp) {
        this.otp = otp;
    }

    public String generateOtp() {
        return String.valueOf(100000 + (int) (Math.random() * 900000));
    }

    public boolean sendMail(String email, String subject, String content) {
        Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.host", this.HOST_NAME);
        props.put("mail.smtp.socketFactory.port", this.SSL_PORT);
        props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.port", this.SSL_PORT);

        Session session = Session.getDefaultInstance(props, new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(SendMail.this.EMAIL, SendMail.this.PASSWORD);
            }
        });

        props.put("mail.smtp.port", this.SSL_PORT);

        Session session = Session.getDefaultInstance(props, new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(SendMail.this.EMAIL, SendMail.this.PASSWORD);
            }
        });

        try {
            MimeMessage message = new MimeMessage(session);
            message.setFrom(new InternetAddress(email));
            message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(email));
            message.setSubject(subject);
            message.setContent(content, "text/html; charset=utf-8");

            Transport.send(message);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }

    public boolean sendOtp(String email) {
        String subject = "Mã dùng một lần của bạn";
        setOtp(generateOtp());
        String content = otp;
        return sendMail(email, subject, content);
    }

    public boolean sendNewPassword(String email, String newPassword) {
        String subject = "Mật khẩu mới của bạn";
        String content = newPassword;
        return sendMail(email, subject, content);
    }
}

```

#### 4.1.4 Mã hoá mật khẩu

Hệ thống sẽ mã hóa mật khẩu trước khi lưu vào cơ sở dữ liệu trước khi tạo mới hoặc có sự thay đổi về mật khẩu.

```
package util;

import java.security.MessageDigest;

public class HashPassword {
    public static String hashPassword(String password) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            byte[] hash = md.digest(password.getBytes());
            StringBuilder sb = new StringBuilder();
            for (byte b : hash) {
                sb.append(String.format("%02x", b));
            }
            return sb.toString();
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

#### 4.1.5 Tạo mật khẩu ngẫu nhiên

```
package util;

public class PasswordGenerator {
    private static final String CHARS = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    private static final int PASSWORD_LENGTH = 12;

    public static String generatePassword() {
        StringBuilder password = new StringBuilder(PASSWORD_LENGTH);

        for (int i = 0; i < PASSWORD_LENGTH; i++) {
            int index = (int) (Math.random() * CHARS.length());
            password.append(CHARS.charAt(index));
        }

        return password.toString();
    }
}
```

Tạo ngẫu nhiên mật khẩu để cập nhật và gửi về email trong trường hợp người dùng yêu cầu lấy lại mật khẩu.

#### 4.1.6 Tạo mã hoá đơn ngẫu nhiên

Hệ thống tạo ngẫu nhiên một chuỗi làm mã hóa đơn trong trường hợp tạo mới hóa đơn.

```

package util;

public class BillIdGenerator {
    private static final String CHARS = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    private static final int BILLID_LENGTH = 8;

    public static String generateBillId() {
        StringBuilder billId = new StringBuilder(BILLID_LENGTH);

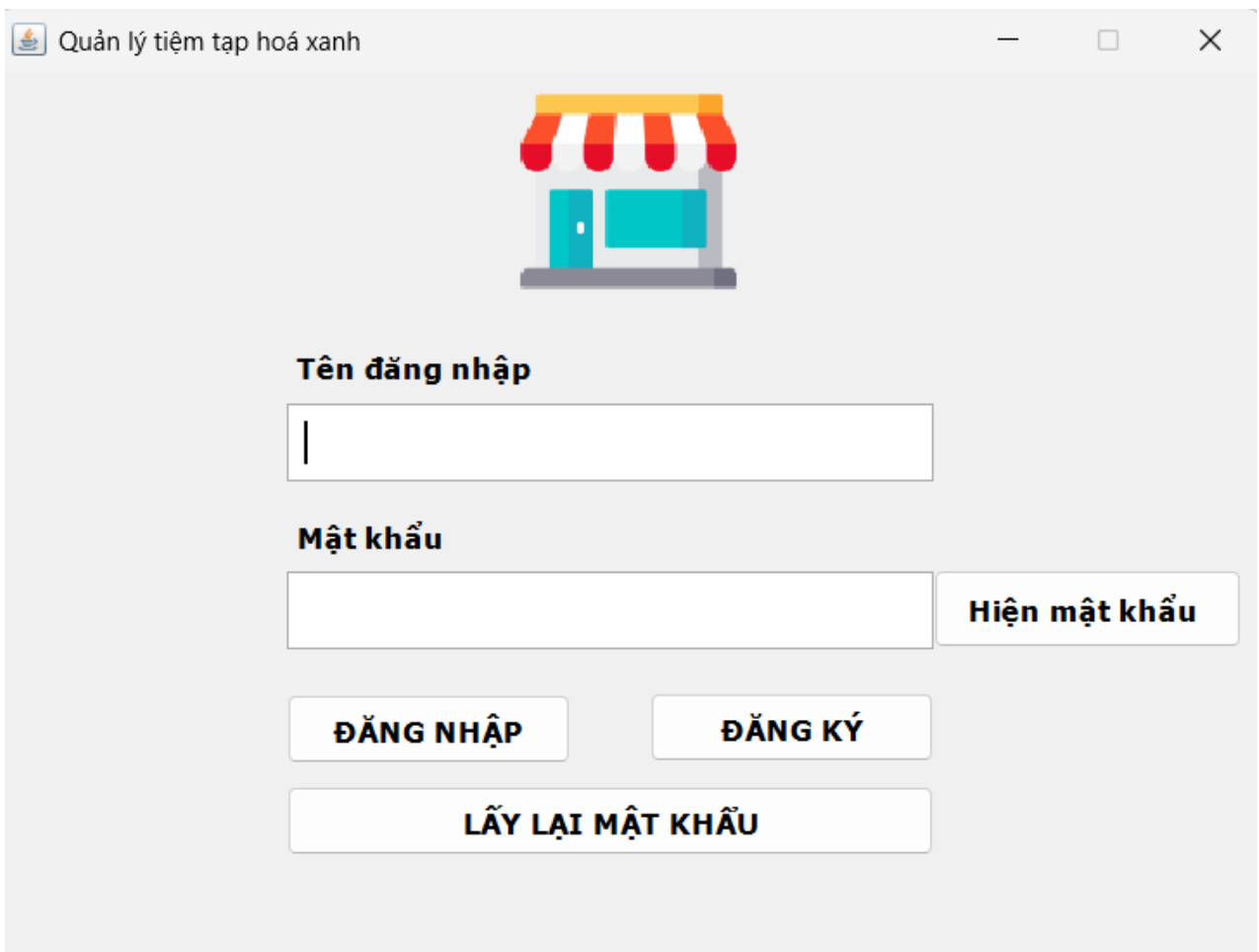
        for (int i = 0; i < BILLID_LENGTH; i++) {
            int index = (int) (Math.random() * CHARS.length());
            billId.append(CHARS.charAt(index));
        }

        return billId.toString();
    }
}


```

## 4.2 Chức năng đăng nhập

### Thiết kế giao diện



Quản lý tiệm tạp hoá xanh



**Tên đăng nhập**

**Mật khẩu**

**Hiện mật khẩu**

**ĐĂNG NHẬP** **ĐĂNG KÝ**

**LẤY LẠI MẬT KHẨU**

Hình XI: Giao diện đăng nhập

### Mô tả chi tiết

Màn hình giao diện đăng nhập bao gồm: tên đăng nhập, mật khẩu; các nút đăng nhập, đăng ký, lấy lại mật khẩu, hiện mật khẩu. Hệ thống không cho phép bỏ trống các trường dữ liệu.

```

JButton loginBtn = new JButton("ĐĂNG NHẬP");
loginBtn.setBounds(0, 1, 131, 32);
panel.add(loginBtn);
loginBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (!FormUtils.ValidateForm(mainPanel)) {
            JOptionPane.showMessageDialog(LoginView.this, "Vui lòng nhập đầy đủ thông tin", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }
    }
});

```

Hệ thống kiểm tra tài khoản và mật khẩu trong cơ sở dữ liệu, cũng như kiểm tra tài khoản đã được xác minh hay chưa.

```

String username = usernameField.getText();
String password = passwordField.getText();

try {
    User user = userDao.get(u -> u.getUsername().equals(username));

    if (user == null || !user.getPassword().equals(HashPassword.hashPassword(password))) {
        JOptionPane.showMessageDialog(LoginView.this, "Tên đăng nhập hoặc mật khẩu không chính xác.",
            "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    if (!user.isVerify()) {
        JOptionPane.showMessageDialog(LoginView.this,
            "Tài khoản của bạn không có quyền truy cập. Vui lòng liên hệ Admin.", "Error",
            JOptionPane.ERROR_MESSAGE);
        return;
    }

    JOptionPane.showMessageDialog(LoginView.this, "Đăng nhập thành công");
    dispose();
    new Home(user);
} catch (ClassNotFoundException e1) {
    e1.printStackTrace();
} catch (IOException e1) {
    e1.printStackTrace();
}

});

```

Người dùng có thể chuyển sang giao diện đăng ký bằng cách nhấn nút đăng ký.

```

JButton signupBtn = new JButton("ĐĂNG KÝ");
signupBtn.setBounds(168, 0, 131, 32);
panel.add(signupBtn);
signupBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        FormUtils.resetForm(mainPanel);
        dispose();
        new SignupView();
    }
});
signupBtn.setFont(new Font("Tahoma", Font.BOLD, 14));

```

Người dùng hiển thị mật khẩu bằng cách chọn nút “Hiện thị mật khẩu”



```

isHide = true;
JButton togglePassBtn = new JButton("Hiện mật khẩu");
togglePassBtn.setBackground(new Color(255, 255, 255));
togglePassBtn.setHorizontalAlignment(SwingConstants.LEFT);
togglePassBtn.setBounds(434, 120, 142, 36);
loginPanel.add(togglePassBtn);
togglePassBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (isHide) {
            ((JPasswordField) passwordField).setEchoChar((char) 0);
        } else {
            ((JPasswordField) passwordField).setEchoChar('•');
        }
        isHide = !isHide;
    }
});

```

### 4.3 Chức năng đăng kí

#### Thiết kế giao diện

The screenshot shows a Java Swing window titled "Đăng ký tài khoản" (Register Account). The window has a standard title bar with minimize, maximize, and close buttons. The main content area is light gray and contains the following elements:

- Tên đăng nhập** (Username): A text input field.
- Mật khẩu** (Password): A text input field with a button labeled **Hiện mật khẩu** (Show Password) to its right.
- Nhập lại mật khẩu** (Repeat Password): A text input field with a button labeled **Hiện mật khẩu** (Show Password) to its right.
- Email**: A text input field.
- Mã xác nhận** (Verification Code): A text input field with a button labeled **GỬI MÃ** (Send Code) to its right.
- At the bottom, there are two buttons: **ĐĂNG KÝ** (Register) and **ĐĂNG NHẬP** (Login).

Hình XII: Giao diện đăng ký

Màn hình giao diện đăng ký bao gồm: tên đăng nhập, mật khẩu, nhập lại mật khẩu, email, mã xác nhận; các nút Gửi mã, Đăng ký, Đăng nhập. Hệ thống gửi mã và kiểm tra email khi nhấn nút Gửi mã, không cho phép bỏ trống các trường dữ liệu.

```
JButton otpBtn = new JButton("GỬI MÃ");
otpBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String username = usernameField.getText();
        String email = emailField.getText();

        if (email.isEmpty()) {
            JOptionPane.showMessageDialog(SignupView.this, "Vui lòng điền email.", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        try {
            if (userDAO.isUserExist(username, email)) {
                JOptionPane.showMessageDialog(SignupView.this,
                    "Đã tồn tại tài khoản với tên tài khoản hoặc email này. Vui lòng thử lại.", "Error",
                    JOptionPane.ERROR_MESSAGE);
                return;
            }
        } catch (ClassNotFoundException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }

        if (!sm.sendOtp(emailField.getText())) {
            JOptionPane.showMessageDialog(SignupView.this,
                "Có lỗi trong quá trình lấy mã xác nhận. Vui lòng thử lại", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        JOptionPane.showMessageDialog(SignupView.this, "Mã xác nhận đã được gửi vào email. Vui lòng kiểm tra.");
    }
});
otpBtn.setFont(new Font("Tahoma", Font.BOLD, 14));
otpBtn.setBounds(478, 354, 148, 37);
signupPanel.add(otpBtn);
```

Hệ thống kiểm tra mật khẩu, nhập lại mật khẩu, kiểm tra mã xác nhận, kiểm tra xem tên đăng nhập hay email đã tồn tại trong cơ sở dữ liệu hay chưa và tạo người dùng mới.

```

String username = usernameField.getText();
String password = passwordField.getText();
String rePassword = rePasswordField.getText();
String email = emailField.getText();
String otp = otpField.getText();

if (!password.equals(rePassword)) {
    JOptionPane.showMessageDialog(SignupView.this, "Vui lòng kiểm tra lại mật khẩu", "Error",
        JOptionPane.ERROR_MESSAGE);
    return;
}

if (!otp.equals(sm.getOtp())) {
    JOptionPane.showMessageDialog(SignupView.this, "Vui lòng kiểm tra lại mã xác nhận", "Error",
        JOptionPane.ERROR_MESSAGE);
    return;
}

UserDAO userDAO = new UserDAO();
UserController userController = new UserController(userDAO);
int id;
try {
    id = userController.getAllUsers().size() + 1;
    User user = new User(id, username, email, password, false);
    if (!userDAO.add(user)) {
        JOptionPane.showMessageDialog(SignupView.this,
            "Đã tồn tại tài khoản với tên tài khoản hoặc email này. Vui lòng thử lại.", "Error",
            JOptionPane.ERROR_MESSAGE);
        return;
    }
} catch (ClassNotFoundException | IOException e1) {
    e1.printStackTrace();
}

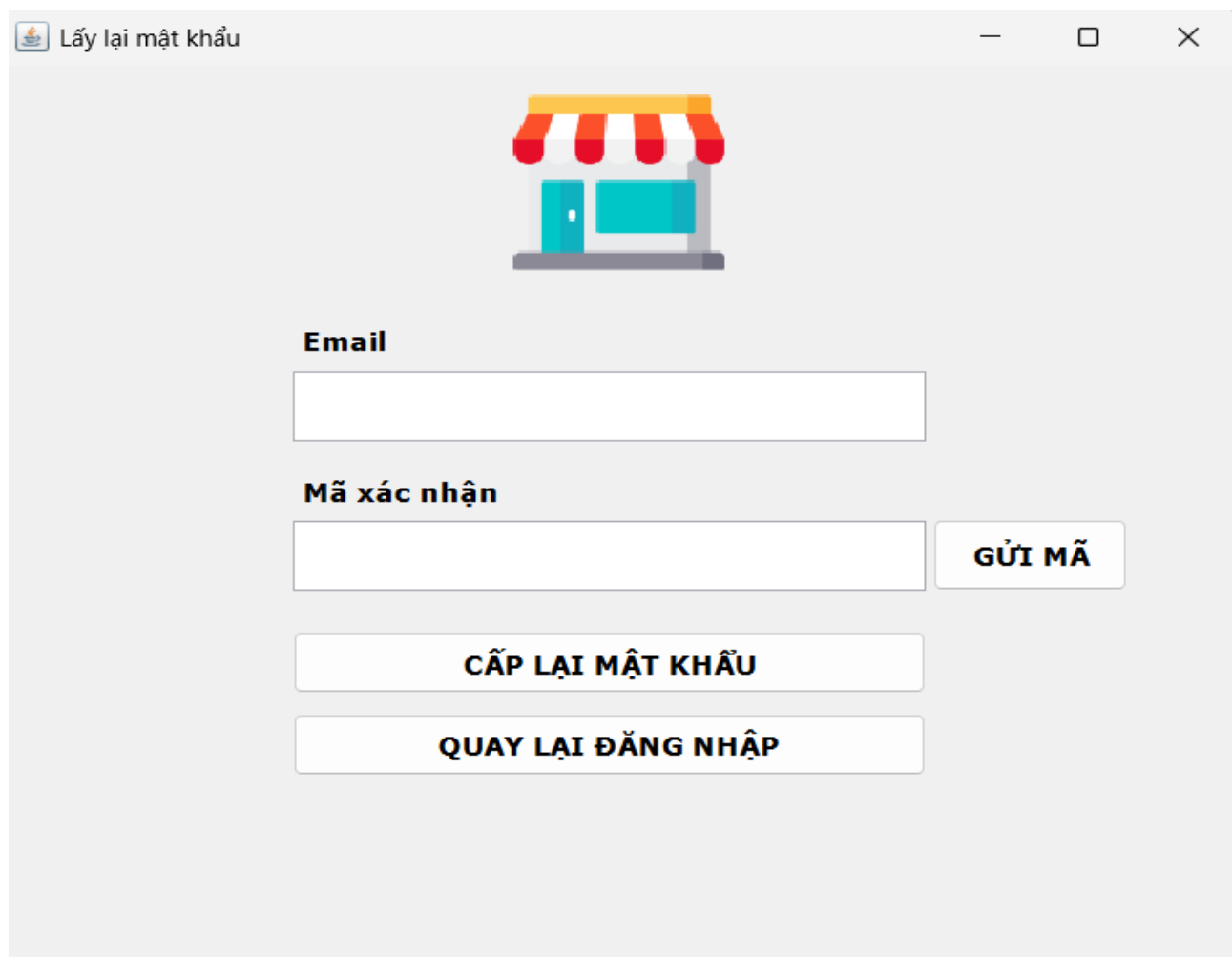
JOptionPane.showMessageDialog(SignupView.this, "Đăng ký thành công");

FormUtils.resetForm(mainPanel);
dispose();
new LoginView();
}
});

```

## 4.4 Chức năng lấy lại mật khẩu

### Thiết kế giao diện



The screenshot shows a web application window titled "Lấy lại mật khẩu" (Reset Password). The window has a standard title bar with minimize, maximize, and close buttons. The main content area features a stylized storefront icon with a red and white striped awning. Below the icon, there are two input fields: one labeled "Email" and another labeled "Mã xác nhận" (Verification Code). To the right of the "Mã xác nhận" field is a button labeled "GỬI MÃ" (Send Code). Below these fields are two large buttons: "CẤP LẠI MẬT KHẨU" (Reset Password) and "QUAY LẠI ĐĂNG NHẬP" (Return to Login).

*Hình XIII: Giao diện quên mật khẩu*

Màn hình giao diện lấy lại mật khẩu bao gồm: các trường email, mã xác nhận; các nút gửi mã xác nhận, cấp lại mật khẩu, quay lại đăng nhập. Hệ thống kiểm tra các thông tin và thông báo lỗi nếu thiếu bất kỳ trường thông tin nào.

```

JButton otpBtn = new JButton("GỬI MÃ");
otpBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String email = emailField.getText();

        if (email.isEmpty()) {
            JOptionPane.showMessageDialog(ResetPasswordView.this, "Vui lòng điền email.", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (!sm.sendOtp(emailField.getText())) {
            JOptionPane.showMessageDialog(ResetPasswordView.this,
                "Có lỗi trong quá trình lấy mã xác nhận. Vui lòng thử lại", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        JOptionPane.showMessageDialog(ResetPasswordView.this,
            "Mã xác nhận đã được gửi vào email. Vui lòng kiểm tra.");
    }
});

```

Hệ thống kiểm tra các trường nhập liệu, kiểm tra xem email có tồn tại trong cơ sở dữ liệu không và kiểm tra mã xác nhận.

```

JButton resetPassBtn = new JButton("CẬP LẠI MẬT KHẨU");
resetPassBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String email = emailField.getText();
        String otp = otpField.getText();

        if (!FormUtils.ValidateForm(mainPanel)) {
            JOptionPane.showMessageDialog(ResetPasswordView.this, "Vui lòng nhập đầy đủ thông tin", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        try {
            User user = userDao.get(u -> u.getEmail().equals(email));
            if (user == null) {
                JOptionPane.showMessageDialog(ResetPasswordView.this, "Email không tồn tại. Vui lòng thử lại.",
                    "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }
        } catch (ClassNotFoundException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        }

        if (!otp.equals(sm.getOtp())) {
            JOptionPane.showMessageDialog(ResetPasswordView.this, "Vui lòng kiểm tra lại mã xác nhận", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        String password = PasswordGenerator.generatePassword();
    }
});

```

Hệ thống tạo ngẫu nhiên mật khẩu, cập nhật vào cơ sở dữ liệu và gửi về email người dùng.

```

String password = PasswordGenerator.generatePassword();

if (!sm.sendNewPassword(email, password)) {
    JOptionPane.showMessageDialog(ResetPasswordView.this,
        "Có lỗi trong quá trình lấy lại mật khẩu. Vui lòng thử lại", "Error",
        JOptionPane.ERROR_MESSAGE);
    return;
}

try {
    User user = userDao.get(u -> u.getEmail().equals(email));
    user.setPassword(HashPassword.hashPassword(password));
    if (!userDAO.update(user)) {
        JOptionPane.showMessageDialog(ResetPasswordView.this,
            "Có lỗi trong quá trình lấy lại mật khẩu. Vui lòng thử lại", "Error",
            JOptionPane.ERROR_MESSAGE);
        return;
    }
} catch (ClassNotFoundException e1) {
    e1.printStackTrace();
} catch (IOException e1) {
    e1.printStackTrace();
}

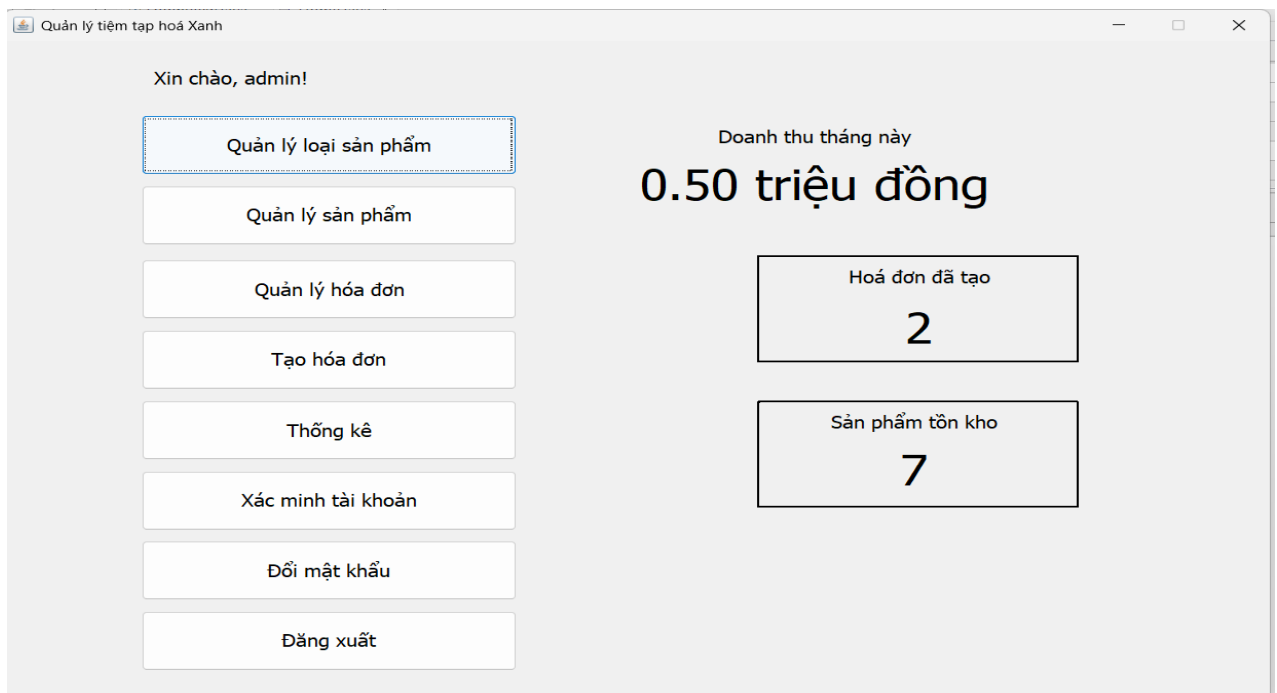
JOptionPane.showMessageDialog(ResetPasswordView.this,
    "Mật khẩu mới đã được gửi về email. Vui lòng kiểm tra");

FormUtils.resetForm(mainPanel);
dispose();
new LoginView();
});

```

## 4.5 Màn hình chính

### Thiết kế giao diện



Hình XIV: Giao diện màn hình chính

Màn hình chính có các lựa chọn gồm Quản lý loại sản phẩm, Quản lý sản phẩm, Quản lý hóa đơn, Tạo hóa đơn, Xác minh tài khoản, Thống kê, Đổi mật khẩu, Đăng xuất để mở cửa sổ thực hiện các chức năng tương ứng. Hiện thị thống kê số sản phẩm, số hóa đơn, doanh thu tháng này.

Chức năng đăng xuất:

```
JButton logoutBtn = new JButton("Đăng xuất");
logoutBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int option = JOptionPane.showConfirmDialog(Home.this, "Bạn có muốn đăng xuất không?");
        if (option == 0) {
            dispose();
            new LoginView();
        }
    }
});
```

Chức năng quản lý loại sản phẩm

```
JButton categoryBtn = new JButton("Quản lý loại sản phẩm");
categoryBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        dispose();
        new CategoryManagementView(user);
    }
});
categoryBtn.setFont(new Font("Tahoma", Font.PLAIN, 16));
categoryBtn.setBounds(107, 250, 292, 56);
mainPanel.add(categoryBtn);
```

Chức năng quản lý hoá đơn

```
JButton billBtn = new JButton("Quản lý hóa đơn");
billBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        dispose();
        new BillManagementView(user);
    }
});
billBtn.setFont(new Font("Tahoma", Font.PLAIN, 16));
billBtn.setBounds(107, 382, 292, 56);
mainPanel.add(billBtn);
```

Chức năng xác minh tài khoản

```
JButton verifyUserBtn = new JButton("Xác minh tài khoản");
verifyUserBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new VerifyUserView(user);
    }
});
verifyUserBtn.setFont(new Font("Tahoma", Font.PLAIN, 16));
verifyUserBtn.setBounds(422, 250, 292, 56);
mainPanel.add(verifyUserBtn);
```

## Chức năng thống kê

```

JButton statBtn = new JButton("Thống kê");
statBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        Map<String, Double> revenue = new TreeMap<String, Double>();

        BillDAO billDAO = new BillDAO();
        BillController billController = new BillController(billDAO, null);
        try {
            List<Bill> bills = billController.getAllBills();

            for (int i = 1; i <= 12; i++) {
                String month = String.format("%02d", i);
                revenue.put(month, revenue.getOrDefault(month, 0.0));
            }

            for (Bill bill : bills) {
                String month = bill.getDate().substring(3, 5);
                revenue.put(month, revenue.getOrDefault(month, 0.0) + bill.getTotal());
            }

            for (Map.Entry<String, Double> entry : revenue.entrySet()) {
                dataset.addValue(entry.getValue(), "Doanh thu (đồng)", entry.getKey());
            }

            JFreeChart barChart = ChartFactory.createBarChart("Thống kê doanh thu", "Tháng", "Doanh thu (đồng)",
                dataset, PlotOrientation.VERTICAL, true, true, false);

            JFrame frame = new JFrame();
            frame.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
            frame.setSize(960, 540);
            frame.getContentPane().add(new ChartPanel(barChart));
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
        } catch (ClassNotFoundException | IOException e1) {
            e1.printStackTrace();
        }
    }
});
statBtn.setFont(new Font("Tahoma", Font.PLAIN, 16));
statBtn.setBounds(107, 515, 292, 56);
mainPanel.add(statBtn);
```

## Chức năng đổi mật khẩu

```

JButton changePassBtn = new JButton("Đổi mật khẩu");
changePassBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new ChangePasswordView(user);
    }
});

changePassBtn.setFont(new Font("Tahoma", Font.PLAIN, 16));
changePassBtn.setBounds(422, 316, 292, 56);
mainPanel.add(changePassBtn);
```

## Chức năng tạo hoá đơn



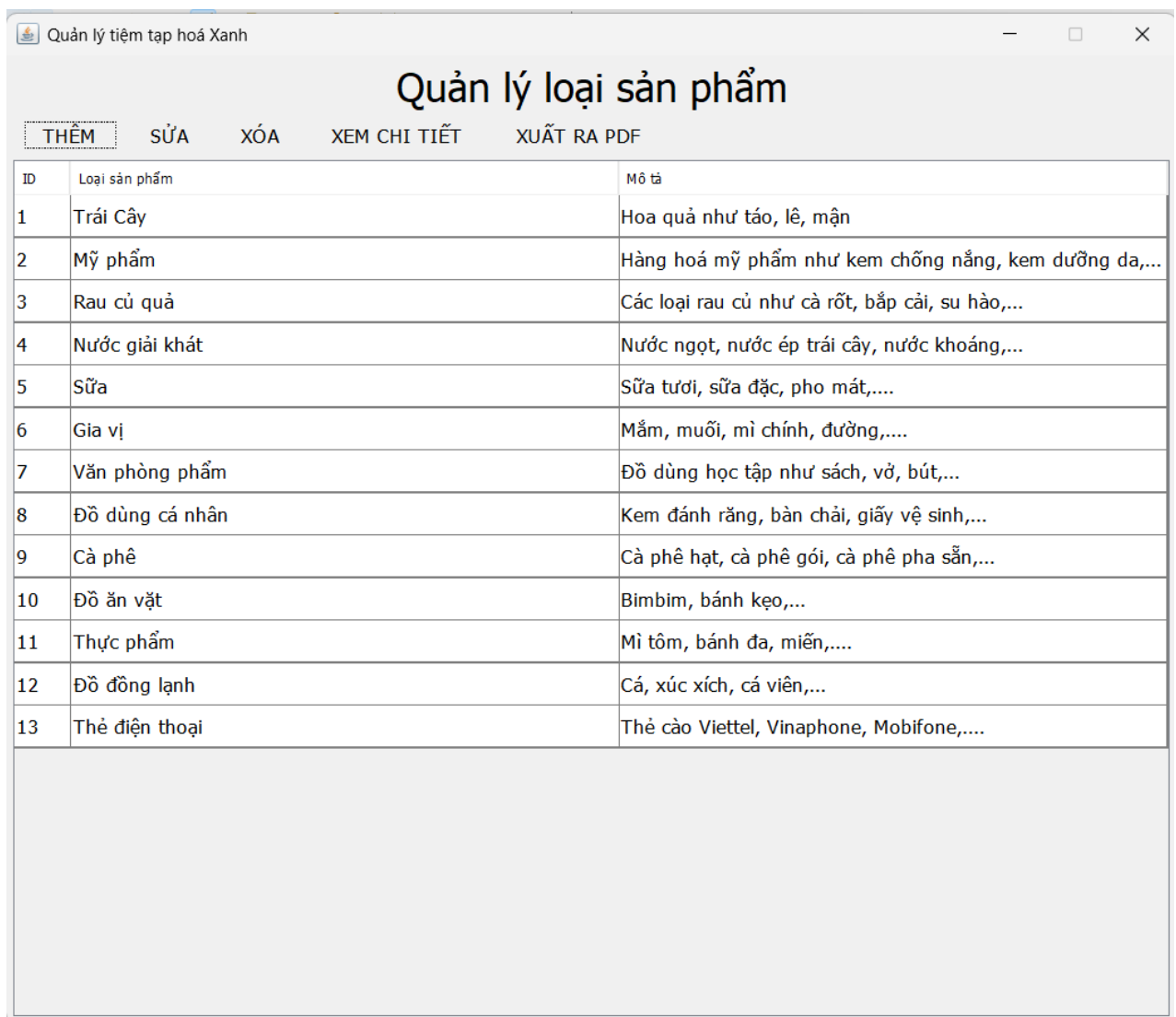
```

JButton addBillBtn = new JButton("Tạo hóa đơn");
addBillBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        dispose();
        new BillView(billController, user);
    }
});
addBillBtn.setFont(new Font("Tahoma", Font.PLAIN, 16));
addBillBtn.setBounds(107, 448, 292, 56);
mainPanel.add(addBillBtn);

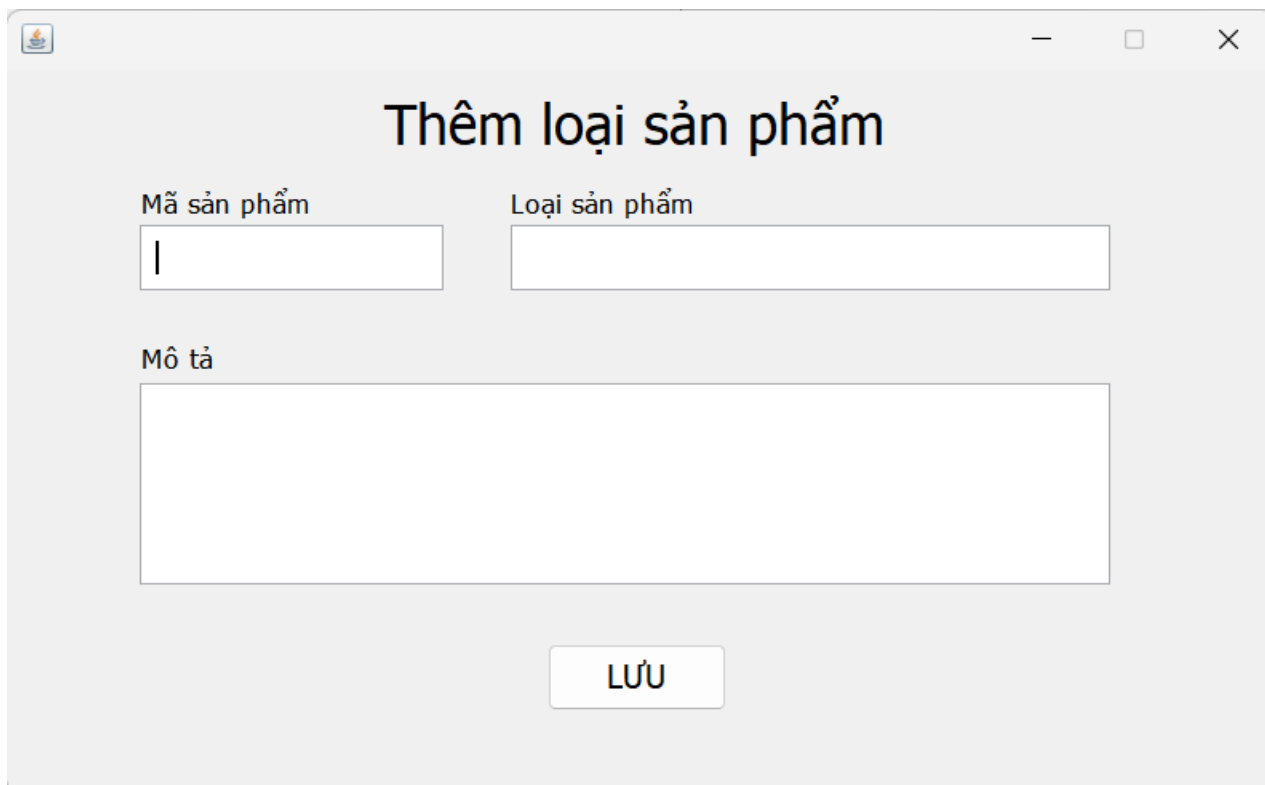
```

## 4.6 Chức năng quản lý loại sản phẩm

### Thiết kế giao diện



Hình XV: Giao diện quản lý sản phẩm



Hình XVI: Giao diện thêm loại sản phẩm

Chức năng “Thêm”:

```

JButton addBtn = new JButton("THÊM");
addBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new CategoryView(categoryController, null, false);
    }
});

addBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(addBtn);
addBtn.setOpaque(false);
addBtn.setContentAreaFilled(false);
addBtn.setBorderPainted(false);
ButtonHover.addButtonHover(addBtn);

```

Chức năng “Sửa”:

```

JButton editBtn = new JButton("SỬA");
editBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Category category = getSelectedCategory();
        if (category == null) {
            JOptionPane.showMessageDialog(CategoryManagementView.this, "Vui lòng chọn loại sản phẩm", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }
        new CategoryView(categoryController, category, true);
    }
});
editBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(editBtn);
editBtn.setOpaque(false);
editBtn.setContentAreaFilled(false);
editBtn.setBorderPainted(false);
ButtonHover.addButtonHover(editBtn);

```

## Xóa loại sản phẩm:

```
JButton deleteBtn = new JButton("XÓA");
deleteBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Category category = getSelectedCategory();
        if (category == null) {
            JOptionPane.showMessageDialog(CategoryManagementView.this, "Vui lòng chọn loại sản phẩm", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        int option = JOptionPane.showConfirmDialog(CategoryManagementView.this,
            "Bạn có chắc chắn muốn xóa không?");

        if (option == 0) {
            try {
                if (!categoryController.deleteCategory(category)) {
                    JOptionPane.showMessageDialog(CategoryManagementView.this, "Xóa thất bại", "Error",
                        JOptionPane.ERROR_MESSAGE);
                    return;
                }
            } catch (HeadlessException | ClassNotFoundException | IOException e1) {
                e1.printStackTrace();
            }

            try {
                updateCategoryTable(categoryController.getAllCategories());
            } catch (ClassNotFoundException | IOException e1) {
                e1.printStackTrace();
            }
            JOptionPane.showMessageDialog(CategoryManagementView.this, "Xóa thành công");
        }
    }
});
deleteBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(deleteBtn);
deleteBtn.setOpaque(false);
deleteBtn.setContentAreaFilled(false);
deleteBtn.setBorderPainted(false);

ButtonHover.addButtonHover(deleteBtn);
```

## Chức năng xem chi tiết :

```
JButton detailBtn = new JButton("XEM CHI TIẾT");
detailBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Category category = getSelectedCategory();
        if (category == null) {
            JOptionPane.showMessageDialog(CategoryManagementView.this, "Vui lòng chọn loại sản phẩm", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        new ProductByCategoryView(category);
    }
});
detailBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(detailBtn);
detailBtn.setOpaque(false);
detailBtn.setContentAreaFilled(false);
detailBtn.setBorderPainted(false);

ButtonHover.addButtonHover(detailBtn);

JButton exportBtn = new JButton("XUẤT RA PDF");
exportBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        MessageFormat header = new MessageFormat(titleLabel.getText());
        MessageFormat footer = new MessageFormat("Tiệm tạp hoá Xanh");
        try {
            PrintRequestAttributeSet set = new HashPrintRequestAttributeSet();
            set.add(OrientationRequested.PORTRAIT);
            table.print(JTable.PrintMode.FIT_WIDTH, header, footer, true, set, true);
            JOptionPane.showMessageDialog(null, "Print successfully");
        } catch (Exception e1) {
            e1.printStackTrace();
        }
    }
});
exportBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(exportBtn);
exportBtn.setOpaque(false);
exportBtn.setContentAreaFilled(false);
exportBtn.setBorderPainted(false);
ButtonHover.addButtonHover(exportBtn);
```

In thông tin ra file pdf:

```
JButton exportBtn = new JButton("XUẤT RA PDF");
exportBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        MessageFormat header = new MessageFormat(titleLabel.getText());
        MessageFormat footer = new MessageFormat("Tiệm tạp hoá Xanh");
        try {
            PrintRequestAttributeSet set = new HashPrintRequestAttributeSet();
            set.add(OrientationRequested.PORTRAIT);
            table.print(JTable.PrintMode.FIT_WIDTH, header, footer, true, set, true);
            JOptionPane.showMessageDialog(null, "Print successfully");
        } catch (Exception e1) {
            e1.printStackTrace();
        }
    }
});
exportBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(exportBtn);
exportBtn.setOpaque(false);
exportBtn.setContentAreaFilled(false);
exportBtn.setBorderPainted(false);
ButtonHover.addButtonHover(exportBtn);
```

## 4.7 Chức năng quản lý sản phẩm

### Thiết kế giao diện

ID	Tên sản phẩm	Loại sản phẩm	Đơn giá	Số lượng
1	Táo	Trái Cây	10000.0	10
2	Su hào	Rau củ quả	12000.0	30
3	Coca cola	Nước giải khát	15000.0	15
4	Vinamilk (48 hộp)	Sữa	250000.0	15
5	Cà phê sữa NetsCafe 3 in 1 800g	Cà phê	152000.0	30
6	Bột canh Vifon gói 450g	Gia vị	15000.0	38
7	Dầu gội Head & Shoulder	Mỹ phẩm	149000.0	8

Hình XVII: Giao diện quản lý sản phẩm

*Hình XVIII: Giao diện thêm sản phẩm*

Chức năng “Thêm”:

```

JButton addBtn = new JButton("THÊM");
addBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new ProductView(productController, null, false);
    }
});

addBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(addBtn);
addBtn.setOpaque(false);
addBtn.setContentAreaFilled(false);
addBtn.setBorderPainted(false);
ButtonHover.addButtonHover(addBtn);

```

## Chức năng sửa:

```
JButton editBtn = new JButton("SỬA");
editBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Product product = getSelectedProduct();
        if (product == null) {
            JOptionPane.showMessageDialog(ProductManagementView.this, "Vui lòng chọn loại sản phẩm", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }
        new ProductView(productController, product, true);
    }
});
editBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(editBtn);
editBtn.setOpaque(false);
editBtn.setContentAreaFilled(false);
editBtn.setBorderPainted(false);
ButtonHover.addButtonHover(editBtn);
```

## Xoá sản phẩm

```
JButton deleteBtn = new JButton("XÓA");
deleteBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Product product = getSelectedProduct();
        if (product == null) {
            JOptionPane.showMessageDialog(ProductManagementView.this, "Vui lòng chọn loại sản phẩm", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        int option = JOptionPane.showConfirmDialog(ProductManagementView.this,
            "Bạn có chắc chắn muốn xóa không?");

        if (option == 0) {
            try {
                if (!productController.deleteProduct(product)) {
                    JOptionPane.showMessageDialog(ProductManagementView.this, "Xóa thất bại", "Error",
                        JOptionPane.ERROR_MESSAGE);
                    return;
                }
            } catch (HeadlessException | ClassNotFoundException | IOException e1) {
                e1.printStackTrace();
            }

            try {
                updateProductTable(productController.getAllProducts());
            } catch (ClassNotFoundException | IOException e1) {
                e1.printStackTrace();
            }
            JOptionPane.showMessageDialog(ProductManagementView.this, "Xóa thành công");
        }
    }
});
deleteBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(deleteBtn);
deleteBtn.setOpaque(false);
deleteBtn.setContentAreaFilled(false);
deleteBtn.setBorderPainted(false);
ButtonHover.addButtonHover(deleteBtn);
```

## Sắp xếp danh sách sản phẩm theo giá:

```
JLabel sortLabel = new JLabel("Sắp xếp theo giá");
sortLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
sortPanel.add(sortLabel, BorderLayout.NORTH);

sortComboBox = new JComboBox<String>();
sortComboBox
    .setModel(new DefaultComboBoxModel<String>(new String[] { "Không sắp xếp", "Tăng dần", "Giảm dần" }));
sortComboBox.setFont(new Font("Tahoma", Font.PLAIN, 16));
sortComboBox.setBorder(BorderFactory.createCompoundBorder(sortComboBox.getBorder(),
    BorderFactory.createEmptyBorder(5, 5, 5, 5)));
sortPanel.add(sortComboBox, BorderLayout.SOUTH);

sortComboBox.addActionListener(new ActionListener() {
    private Timer sortTimer = new Timer(500, e -> {
        int choose = sortComboBox.getSelectedIndex();
        if (choose == 0)
            return;
        boolean isINC = choose == 1 ? true : false;
        productController.sortProductsByPrice(getDataFromTable(), isINC);
    });
});

@Override
public void actionPerformed(ActionEvent e) {
    if (sortTimer.isRunning()) {
        sortTimer.restart();
    } else {
        sortTimer.start();
    }
}
```

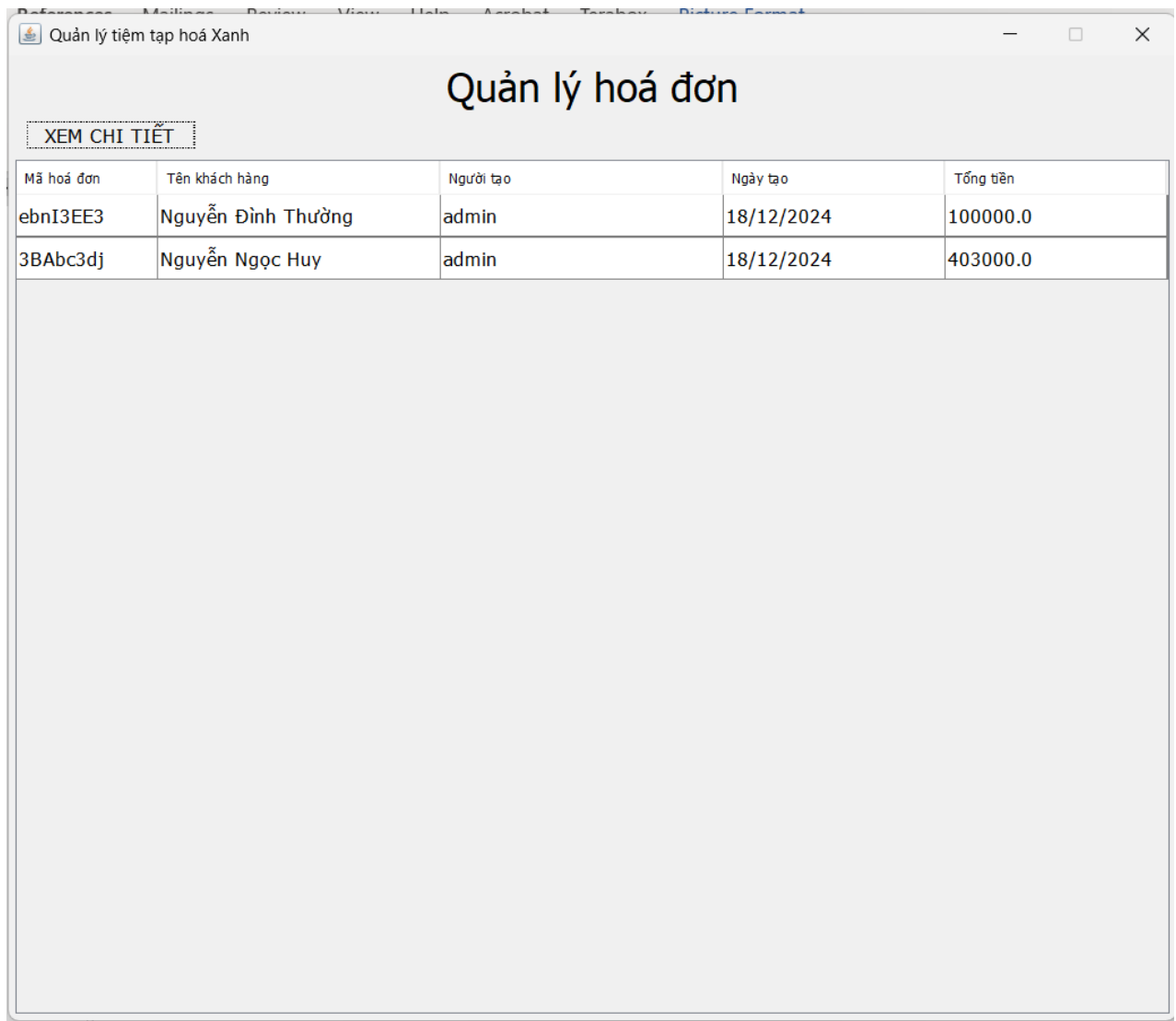
## Xuất danh sách sản phẩm ra PDF:

```
JButton exportBtn = new JButton("XUẤT RA PDF");
exportBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        MessageFormat header = new MessageFormat(titleLabel.getText());
        MessageFormat footer = new MessageFormat("Tiệm tạp hoá Xanh");
        try {
            PrintRequestAttributeSet set = new HashPrintRequestAttributeSet();
            set.add(OrientationRequested.PORTRAIT);
            table.print(JTable.PrintMode.FIT_WIDTH, header, footer, true, set, true);
            JOptionPane.showMessageDialog(null, "Print successfully");
        } catch (Exception e1) {
            e1.printStackTrace();
        }
    }
});
exportBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(exportBtn);
exportBtn.setOpaque(false);
exportBtn.setContentAreaFilled(false);
exportBtn.setBorderPainted(false);
ButtonHover.addButtonHover(exportBtn);
```

## 4.8 Chức năng quản lý hoá đơn

### Thiết kế giao diện





Hình XIX: Giao diện quản lý hoá đơn

Code chương trình:

```
public BillManagementView(User user) {
    billDAO = new BillDAO();
    billController = new BillController(billDAO, this);

    addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(WindowEvent e) {
            dispose();
            new Home(user);
        }
    });
}
```

```

setResizable(false);
setTitle("Quản lý tiệm tạp hoá Xanh");
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
setBounds(100, 100, 840, 720);
mainPanel = new JPanel();
mainPanel.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(mainPanel);
mainPanel.setLayout(new BorderLayout(0, 0));

JLabel titleLabel = new JLabel("Quản lý hoá đơn", SwingConstants.CENTER);
titleLabel.setFont(new Font("Tahoma", Font.PLAIN, 28));
mainPanel.add(titleLabel, BorderLayout.NORTH);

JPanel panel = new JPanel();
mainPanel.add(panel, BorderLayout.CENTER);
panel.setLayout(new BorderLayout(0, 0));

JPanel headerPanel = new JPanel();
panel.add(headerPanel, BorderLayout.NORTH);
FlowLayout fl_headerPanel = new FlowLayout(FlowLayout.LEFT, 5, 5);
headerPanel.setLayout(fl_headerPanel);

JButton detailBtn = new JButton("XEM CHI TIẾT");
detailBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Bill bill = getSelectedBill();
        if (bill == null) {
            JOptionPane.showMessageDialog(BillManagementView.this, "Vui lòng chọn hoá đơn", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }
        new BillDetailView(bill);
    }
});

detailBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
headerPanel.add(detailBtn);
detailBtn.setOpaque(false);
detailBtn.setContentAreaFilled(false);
detailBtn.setBorderPainted(false);

ButtonHover.addButtonHover(detailBtn);

JLabel lblNewLabel = new JLabel("
headerPanel.add(lblNewLabel);

table = new JTable();
table.setModel(new DefaultTableModel(new Object[][] {},
    new String[] { "Mã hoá đơn", "Tên khách hàng",
        "Người tạo", "Ngày tạo", "Tổng tiền" }) {
    private static final long serialVersionUID = 1L;

    @Override
    public boolean isCellEditable(int row, int column) {
        return false;
    }
});

```

## 4.9 Chức năng tạo hoá đơn

Thiết kế giao diện

**Tạo hoá đơn**

Tên khách hàng

**Thêm sản phẩm**

**Danh sách sản phẩm**

ID	Tên sản phẩm	Đơn giá	Số lượng mua
----	--------------	---------	--------------

**Tạo đơn hàng**

Hình XX: Giao diện tạo hoá đơn

Minh họa mã chương trình:

Lựa chọn “Thêm sản phẩm”:

```

JButton addProductBtn = new JButton("Thêm sản phẩm");
addProductBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new ProductManagementView(true, BillView.this);
    }
});
addProductBtn.setFont(new Font("Tahoma", Font.PLAIN, 16));
addProductBtn.setBounds(259, 81, 165, 33);
mainPanel.add(addProductBtn);

```

Lựa chọn “Tạo đơn hàng”:

```

JButton saveBtn = new JButton("Tạo đơn hàng");
saveBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (!FormUtils.ValidateForm(mainPanel)) {
            JOptionPane.showMessageDialog(BillView.this, "Vui lòng nhập đầy đủ thông tin", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        String id;
        while (true) {
            id = BillIdGenerator.generateBillId();
            try {
                if (billController.getBillById(id) == null)
                    break;
            } catch (ClassNotFoundException | IOException e1) {
                e1.printStackTrace();
            }
        }

        String name = nameField.getText();
        int adminId = user.getId();
        Date date = new Date();

        HashMap<Product, Integer> products = new HashMap<Product, Integer>();
        DefaultTableModel dtm = (DefaultTableModel) table.getModel();
        int rowCount = dtm.getRowCount();
        for (int i = 0; i < rowCount; i++) {
            int productId = Integer.parseInt(dtm.getValueAt(i, 0).toString());
            try {
                Product product = productController.getProductById(productId);
                int buyQuantity = Integer.valueOf(String.valueOf(dtm.getValueAt(i, 3)));
                products.put(product, buyQuantity);
                product.setQuantity(product.getQuantity() - buyQuantity);
                productController.updateProduct(product);
            } catch (ClassNotFoundException | IOException e1) {
                e1.printStackTrace();
            }
        }

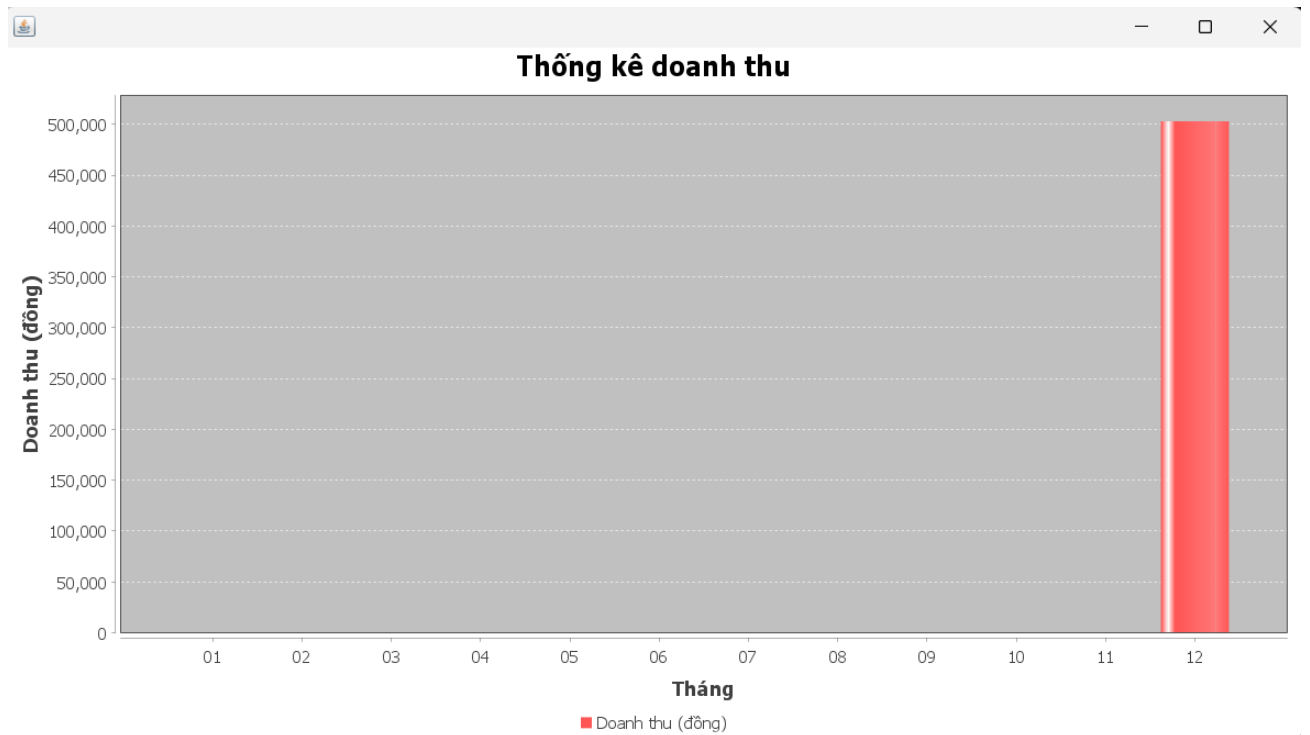
        try {
            if (billController.addBill(new Bill(id, name, adminId, date, products))) {
                JOptionPane.showMessageDialog(BillView.this, "Tạo hóa đơn thành công");
                dispose();
                new Home(user);
            } else {
                JOptionPane.showMessageDialog(BillView.this, "Tạo hóa đơn thất bại", "Error",
                    JOptionPane.ERROR_MESSAGE);
            }
        } catch (ClassNotFoundException | IOException e1) {
            e1.printStackTrace();
        }

        FormUtils.resetForm(mainPanel);
    }
});

```

## 4.10 Chức năng thống kê

### Thiết kế giao diện



*Hình XXI: Giao diện thống kê*

### Minh họa mã chương trình

```

JButton statBtn = new JButton("Thống kê");
statBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        Map<String, Double> revenue = new TreeMap<String, Double>();

        BillDAO billDAO = new BillDAO();
        BillController billController = new BillController(billDAO, null);
        try {
            List<Bill> bills = billController.getAllBills();

            for (int i = 1; i <= 12; i++) {
                String month = String.format("%02d", i);
                revenue.put(month, revenue.getDefault(month, 0.0));
            }

            for (Bill bill : bills) {
                String month = bill.getDate().substring(3, 5);
                revenue.put(month, revenue.getDefault(month, 0.0) + bill.getTotal());
            }

            for (Map.Entry<String, Double> entry : revenue.entrySet()) {
                dataset.addValue(entry.getValue(), "Doanh thu (đồng)", entry.getKey());
            }

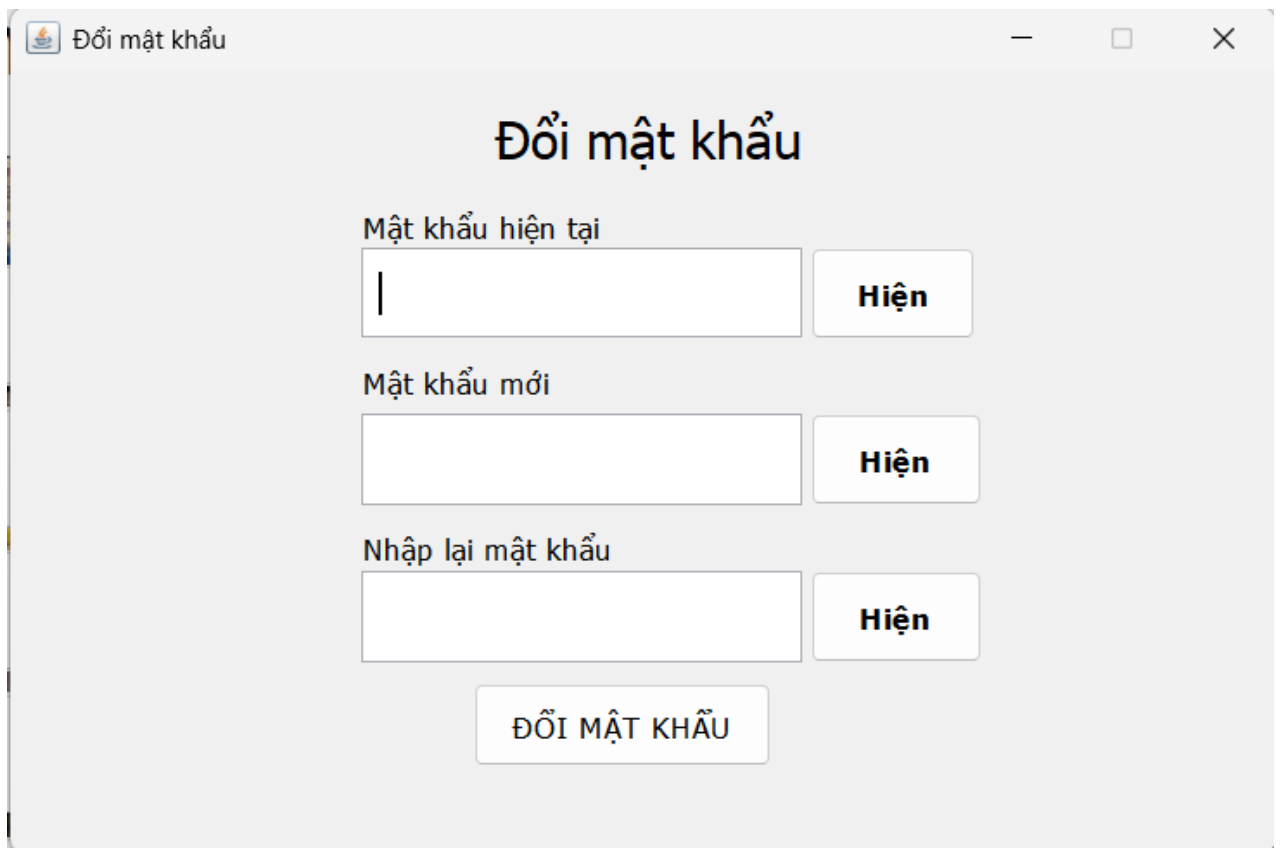
            JFreeChart barChart = ChartFactory.createBarChart("Thống kê doanh thu", "Tháng", "Doanh thu (đồng)",
                dataset, PlotOrientation.VERTICAL, true, true, false);

            JFrame frame = new JFrame();
            frame.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
            frame.setSize(960, 540);
            frame.getContentPane().add(new ChartPanel(barChart));
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
        } catch (ClassNotFoundException | IOException e1) {
            e1.printStackTrace();
        }
    }
});
statBtn.setFont(new Font("Tahoma", Font.PLAIN, 16));
statBtn.setBounds(107, 515, 292, 56);
mainPanel.add(statBtn);

```

## 4.11 Chức năng đổi mật khẩu

### Thiết kế giao diện



The image shows a software window titled "Đổi mật khẩu" (Change Password). It contains three input fields for password entry, each with a corresponding "Hiện" (Show) button to toggle visibility. The fields are labeled "Mật khẩu hiện tại" (Current password), "Mật khẩu mới" (New password), and "Nhập lại mật khẩu" (Repeat password). A large "ĐỔI MẬT KHẨU" (Change Password) button is positioned at the bottom center of the form.

Đổi mật khẩu

Mật khẩu hiện tại

Hiện

Mật khẩu mới

Hiện

Nhập lại mật khẩu

Hiện

ĐỔI MẬT KHẨU

*Hình XXII: Giao diện đổi mật khẩu*

Minh họa mã chương trình:

```

JButton changePassBtn = new JButton("ĐỔI MẬT KHẨU");
changePassBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (!FormUtils.ValidateForm(mainPanel)) {
            JOptionPane.showMessageDialog(ChangePasswordView.this, "Vui lòng nhập đầy đủ thông tin", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        String curPass = curPassField.getText();
        String newPass = newPassField.getText();
        String rePass = rePassField.getText();

        if (!user.getPassword().equals(HashPassword.hashPassword(curPass))) {
            JOptionPane.showMessageDialog(ChangePasswordView.this, "Mật khẩu sai. Vui lòng thử lại.", "Error",
                JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (newPass.equals(curPass)) {
            JOptionPane.showMessageDialog(ChangePasswordView.this, "Mật khẩu mới không được trùng mật khẩu cũ.",
                "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        if (!newPass.equals(rePass)) {
            JOptionPane.showMessageDialog(ChangePasswordView.this, "Mật khẩu không khớp. Vui lòng nhập lại.",
                "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        user.setPassword(HashPassword.hashPassword(newPass));
        try {
            userDAO.update(user);
            JOptionPane.showMessageDialog(ChangePasswordView.this, "Thay đổi mật khẩu thành công.");
        } catch (ClassNotFoundException | IOException e1) {
            e1.printStackTrace();
        }

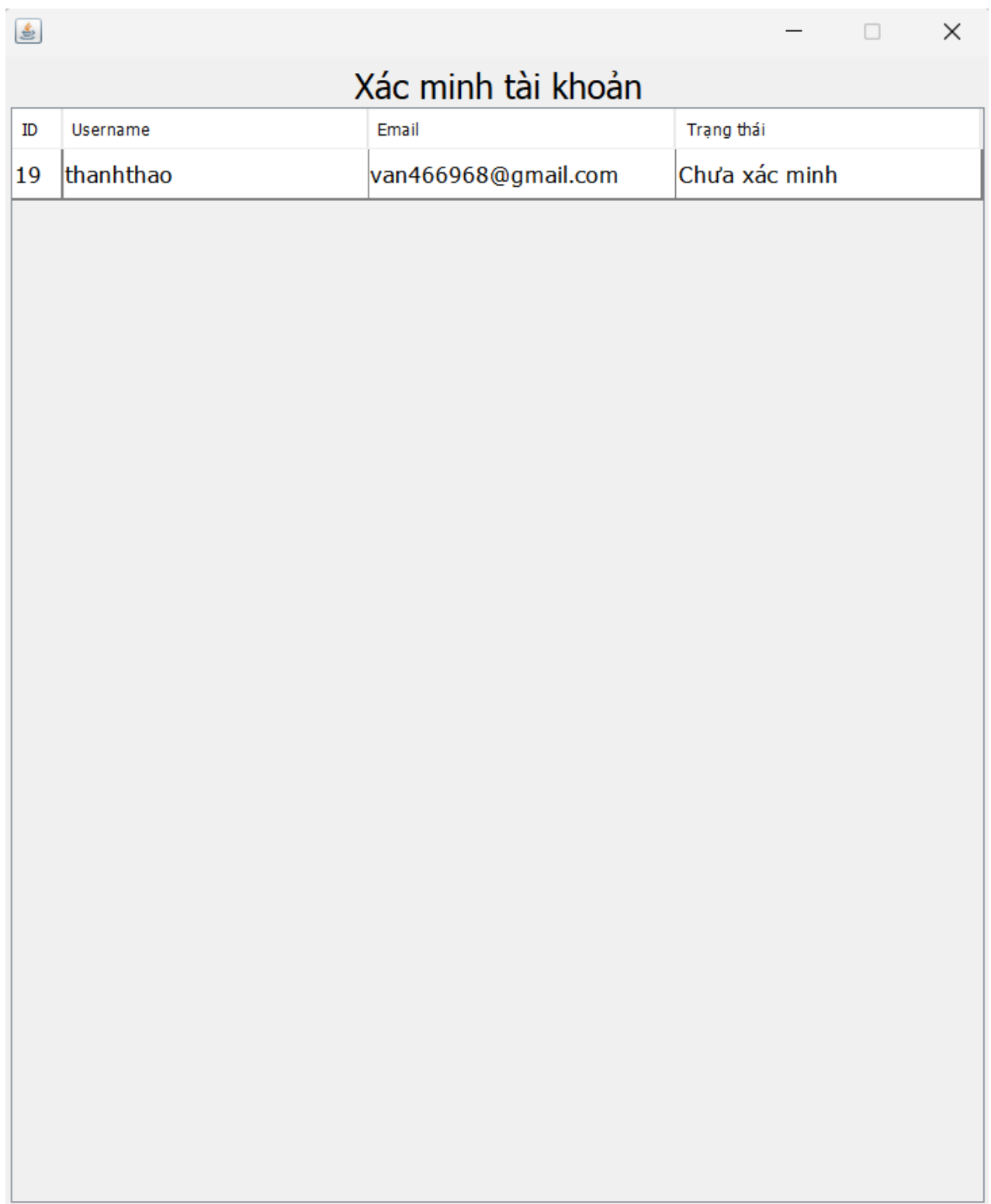
        dispose();
    }
});
changePassBtn.setFont(new Font("Tahoma", Font.PLAIN, 14));
changePassBtn.setBounds(215, 284, 137, 38);
mainPanel.add(changePassBtn);

```

## 4.12 Chức năng xác minh tài khoản

### Thiết kế giao diện





The screenshot shows a web browser window with the title "Xác minh tài khoản". Inside the window, there is a table with the following data:

ID	Username	Email	Trạng thái
19	thanhthao	van466968@gmail.com	Chưa xác minh

The table is followed by a large, empty rectangular area, likely intended for displaying a verification code or additional instructions.

*Hình XXIII: Giao diện xác minh tài khoản*

Minh họa mã chương trình

```

table.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        int row = table.rowAtPoint(e.getPoint());
        int id = Integer.valueOf(String.valueOf(table.getValueAt(row, 0)));
        try {
            User selectedUser = userDao.get(u -> u.getId() == id);

            if (selectedUser != null) {
                int option = JOptionPane.showConfirmDialog(VerifyUserView.this,
                    "Bạn đồng ý xác minh cho tài khoản này?", "Xác minh tài khoản",
                    JOptionPane.YES_NO_OPTION);
                if (option == JOptionPane.YES_OPTION) {
                    selectedUser.setVerify(true);
                    userDao.update(selectedUser);
                    getData();
                }
            }
            getData();
        } catch (ClassNotFoundException | IOException e1) {
            e1.printStackTrace();
        }
    }
});

```

## PHẦN 3: TỔNG KẾT

### a. Tổng kết nội dung

Sau khi thực hiện bài tập lớn, chúng em đã nắm được cái nhìn tổng quan về ngôn ngữ lập trình Java và các tác vụ quan trọng trong một dự án thực tế, như thêm, sửa, xóa, tìm kiếm, sắp xếp,... để hỗ trợ việc quản lý hiệu quả hơn. Chúng em cũng học hỏi được cách xây dựng cơ sở dữ liệu từ việc khảo sát thực tế tại các đơn vị, đồng thời rèn luyện kỹ năng quản lý dự án và áp dụng vào quá trình thực hiện.

#### **Kết quả đạt được:**

- Có khả năng tự xây dựng cơ sở dữ liệu dựa trên nhu cầu thực tế.
- Biết cách cài đặt và phát triển một chương trình cơ bản hoàn chỉnh.
- Hoàn thiện kỹ năng trình bày báo cáo và hướng dẫn sử dụng một cách rõ ràng, dễ hiểu, phục vụ tốt cho người dùng cuối.

### b. Hướng phát triển trong tương lai

Trong quá trình học tập tại trường và thông qua việc tìm hiểu, nghiên cứu trên internet, chúng em đã phát triển một sản phẩm ứng dụng thực tế với các chức năng hữu ích, dễ dàng sử dụng, và phù hợp với nhu cầu thực tiễn. Dự án này đã mang đến cho chúng em cơ hội quý báu để tham gia khảo sát thực tế, hiểu sâu hơn về các nghiệp vụ liên quan, đồng thời vận dụng những kiến thức học được vào thực tiễn.

Chúng em nhận thức rằng, do hạn chế về thời gian và kinh nghiệm, sản phẩm còn nhiều điểm cần cải thiện để hoàn thiện hơn. Chúng em xin gửi lời cảm ơn chân thành tới thầy Hà Mạnh Đào, người đã tận tình hướng dẫn và hỗ trợ chúng em trong suốt quá trình thực hiện dự án. Sự đồng hành và những góp ý quý giá từ thầy chính là động lực lớn, giúp chúng em vượt qua những khó khăn và hoàn thành sản phẩm này.

## TÀI LIỆU THAM KHẢO

<https://www3.ntu.edu.sg/home/ehchua/programming/#SQL>

<https://www3.ntu.edu.sg/home/ehchua/programming/index.html#Java>

<https://www.javatpoint.com/jvm-java-virtual-machine>

<https://github.com/topics/java-swing-application?l=java&o=desc&s=stars>

<https://github.com/janbodnar/Java-Swing-Examples>

[https://github.com/tungtsdev96/OOP20161\\_StoreManager](https://github.com/tungtsdev96/OOP20161_StoreManager)

<https://github.com/quanvuhust/QuanLySinhVien>

<https://github.com/atarw/material-ui-swing>

<https://github.com/topics/java-swing-gui?l=java&o=desc&s=updated>