BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC GIA ĐỊNH**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**TIỂU LUẬN**

**XÂY DỰNG ỨNG DỤNG BÁN ĐIỆN THOẠI**

**DI ĐỘNG**

**MÔN: LẬP TRÌNH ỨNG DỤNG DI ĐỘNG**

**NÂNG CAO**

Giảng viên hướng dẫn: **LÊ HUỲNH PHƯỚC**

Sinh viên thực hiện: **Phạm Văn Phúc**

MSSV: **2104110044**

Lớp: **K15DCPM01**

Ngành: **KĨ THUẬT PHẦN MỀM**

TP. Hồ Chí Minh, tháng 4, năm 2024

**Khoa/Viện: Công Nghệ Thông Tin**

**NHẬN XÉT VÀ CHẤM ĐIỂM CỦA GIẢNG VIÊN**

**TIỂU LUẬN MÔN: LẬP TRÌNH ỨNG DỤNG DI ĐỘNG NÂNG CAO**

1. **Họ và tên sinh viên:** Phạm Văn Phúc
2. **Tên đề tài**: Xây dựng ứng dụng bán điện thoại di động
3. **Nhận xét**:

***Những kết quả đạt được:***

***Những hạn chế:***

1. **Điểm đánh giá** *(theo thang điểm 10, làm tròn đến 0.5):*

Sinh viên:……………………………………………………………………………….

Điểm số: ……….…… Điểm chữ: ……………………………………………………..

|  |  |
| --- | --- |
|  | *TP. HCM, ngày … tháng … năm 20……*  **Giảng viên chấm thi**  *(Ký và ghi rõ họ tên)* |

**MỤC LỤC**

[**LỜI CẢM ƠN** 1](#_Toc164197025)

[**CHƯƠNG 1 TỔNG QUAN** 2](#_Toc164197026)

[**1.1.** **Lý do chọn đề tài** 2](#_Toc164197027)

[**1.2.** **Mục tiêu nghiên cứu** 2](#_Toc164197028)

[**1.3.** **Phạm vi nghiên cứu** 2](#_Toc164197029)

[**CHƯƠNG 2 CƠ SỞ LÝ THUYẾT** 3](#_Toc164197030)

[**2.1. Công cụ sử dụng** 3](#_Toc164197031)

[***2.1.1. Android studio (android app)*** 3](#_Toc164197032)

[***2.1.2. VS Code (web app)*** 3](#_Toc164197033)

[***2.1.3. MongoDB*** 4](#_Toc164197034)

[**2.3. Ngôn ngữ và công nghệ sử dụng bên trong web app** 5](#_Toc164197035)

[***2.3.1. NodeJS (web app)*** 5](#_Toc164197036)

[***2.3.2. ReactJS (fontend - web app)*** 5](#_Toc164197037)

[***2.3.3. Redux (fontend - web app)*** 6](#_Toc164197038)

[***2.3.4. Express (backend – web app)*** 7](#_Toc164197039)

[**2.4. Ngôn ngữ và công nghệ sử dụng bên trong android app** 8](#_Toc164197040)

[***2.4.1. Java*** 8](#_Toc164197041)

[***2.4.2. Kotlin*** 9](#_Toc164197042)

[***2.4.3. Retrofit và thư viện Gson*** 9](#_Toc164197043)

[***2.4.4. Cơ sở dữ liệu cục bộ SQLite*** 10](#_Toc164197044)

[**2.5. Thiết kế cơ sở dữ liệu** 11](#_Toc164197045)

[***2.5.1. Mô hình dữ liệu cho sản phẩm, người dùng và hóa đơn*** 11](#_Toc164197046)

[***2.5.2. Thiết kế cấu trúc cơ sở dữ liệu cho ứng dụng*** 11](#_Toc164197047)

[**CHƯƠNG 3 THIẾT KẾ VÀ TRIỂN KHAI** 13](#_Toc164197048)

[**3.1. Sơ đồ use case** 13](#_Toc164197049)

[***3.1.1. Xác định use case*** 13](#_Toc164197050)

[***3.1.2. Biểu đồ Use Case đăng nhập, đăng xuất*** 14](#_Toc164197051)

[***3.1.3. Biểu đồ Use Case Quản lý sản phẩm*** 15](#_Toc164197052)

[***3.1.4. Biểu đồ Use Case Case quản lý hóa đơn*** 15](#_Toc164197053)

[**3.2. Phân tích yêu cầu** 16](#_Toc164197054)

[***3.2.1. Yêu cầu chức năng*** 16](#_Toc164197055)

[***3.2.2. Yêu cầu phi chức năng*** 16](#_Toc164197056)

[***3.2.3. Phân tích nhu cầu của người dùng*** 17](#_Toc164197057)

[**3.3. Thiết kế giao diện người dùng** 18](#_Toc164197058)

[***3.3.1. Trang đăng nhập*** 18](#_Toc164197059)

[***3.3.2. Trang đăng ký*** 20](#_Toc164197060)

[***3.3.3. Trang chủ*** 22](#_Toc164197061)

[***3.3.4. Giỏ hàng*** 24](#_Toc164197062)

[***3.3.5. Trang quản trị thêm, xoá, sửa sản phẩm, hoá đơn mua hàng.*** 26](#_Toc164197063)

[**3.4. Hiện Thực Ứng Dụng (Android app)** 27](#_Toc164197064)

[***3.4.1. Thiết kế giao diện người dùng*** 27](#_Toc164197065)

[***3.4.2. Xử lý logic, quản lý trạng thái đăng nhập*** 27](#_Toc164197066)

[**3.5. Hiện Thực Ứng Dụng (Web app)** 28](#_Toc164197067)

[***3.5.1. Thiết kế giao diện người dùng ( fontend)*** 28](#_Toc164197068)

[***3.5.2. Xử lý logic (backend)*** 28](#_Toc164197069)

[**CHƯƠNG 4 KẾT LUẬN** 29](#_Toc164197070)

[**5.1. Tóm tắt kết quả** 29](#_Toc164197071)

[**5.2. Đánh Giá và Hướng Phát Triển** 29](#_Toc164197072)

[**5.3. Kết luận** 30](#_Toc164197073)

[**TÀI LIỆU THAM KHẢO** 31](#_Toc164197074)

# **LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành đến thầy Lê Huỳnh Phước về sự hỗ trợ và sự dẫn dắt trong suốt quá trình thực hiện tiểu luận của mình. Sự kiên nhẫn, tận tình và kiến thức sâu rộng của thầy đã giúp em vượt qua những thách thức và hoàn thành dự án một cách thành công.

Thầy đã luôn sẵn lòng chia sẻ kiến thức và kinh nghiệm của mình, giúp em hiểu sâu hơn về chủ đề và phát triển kỹ năng nghiên cứu. Sự động viên và hỗ trợ từ thầy đã giúp em vượt qua những khó khăn và tự tin hơn trong quá trình thực hiện dự án.

Em cũng muốn bày tỏ lòng biết ơn đến sự hướng dẫn và góp ý của thầy trong việc hoàn thiện tiểu luận của mình. Những nhận xét chân thành và xây dựng từ thầy đã giúp em cải thiện và hoàn thiện dự án của mình một cách tốt nhất.

Cuối cùng, em xin chân thành cảm ơn thầy vì sự tận tụy và tâm huyết trong việc giảng dạy và hướng dẫn chúng em. Mong rằng những điều tốt đẹp nhất sẽ đến với thầy và những dự án nghiên cứu sẽ luôn được thúc đẩy và phát triển.

# **CHƯƠNG 1 TỔNG QUAN**

* 1. **Lý do chọn đề tài**

Thị trường điện thoại di động đang phát triển mạnh mẽ và có tiềm năng lớn. Việc tạo ra một ứng dụng bán điện thoại có thể mang lại nhiều cơ hội kinh doanh và thu hút đối tượng khách hàng đa dạng.

Sự phổ biến của điện thoại di động và nhu cầu mua sắm trực tuyến đang gia tăng, đặc biệt là trong thời đại số hóa hiện nay. Việc phát triển một ứng dụng bán điện thoại có thể đáp ứng nhu cầu mua sắm của người tiêu dùng một cách thuận tiện và linh hoạt.

Xây dựng một ứng dụng bán điện thoại đòi hỏi các kỹ năng lập trình, thiết kế giao diện, quản lý cơ sở dữ liệu và triển khai ứng dụng. Việc thực hiện dự án này có thể giúp phát triển và củng cố kỹ năng kỹ thuật trong lĩnh vực phát triển phần mềm di động.

Một ứng dụng bán điện thoại có thể trở thành một công cụ kinh doanh hiệu quả, cung cấp cho bạn cơ hội tiếp cận thị trường mới, tăng doanh số bán hàng và xây dựng thương hiệu của bạn trên nền tảng di động.

* 1. **Mục tiêu nghiên cứu**

Xây dựng một ứng dụng bán điện thoại di động có giao diện người dùng thân thiện, tính năng đa dạng và hiệu suất cao để đáp ứng nhu cầu mua sắm của người dùng.

Nghiên cứu về thị trường và người dùng để hiểu rõ nhu cầu và mong muốn của họ, từ đó thiết kế ứng dụng sao cho phản ánh đúng nhất những yêu cầu này.

Phát triển các biện pháp bảo mật chặt chẽ để bảo vệ thông tin cá nhân và thanh toán của người dùng, đảm bảo an toàn khi sử dụng ứng dụng.

Tối ưu hóa mã nguồn và quản lý cơ sở dữ liệu để cải thiện hiệu suất và ổn định của ứng dụng, đồng thời giảm thiểu thời gian phản hồi và lỗi hệ thống.

Thiết lập các chỉ số hiệu suất và công cụ đo lường để đánh giá và cải thiện hiệu quả của ứng dụng, từ việc thu hút người dùng đến việc tăng doanh số bán hàng.

* 1. **Phạm vi nghiên cứu**

Phát triển ứng dụng: Nghiên cứu và phát triển một ứng dụng di động hoàn chỉnh cho việc mua sắm điện thoại, bao gồm cả giao diện người dùng, tính năng, và hệ thống quản lý dữ liệu.

Thiết kế giao diện người dùng: Tạo ra giao diện người dùng thân thiện, dễ sử dụng và thu hút người dùng, bao gồm cả trải nghiệm mua hàng và thanh toán.

Quản lý cơ sở dữ liệu: Thiết kế và triển khai cơ sở dữ liệu phù hợp để lưu trữ thông tin sản phẩm, thông tin người dùng, và lịch sử giao dịch.

Tối ưu hóa hiệu suất: Nghiên cứu và tối ưu hóa mã nguồn, quản lý tài nguyên, và tăng cường hiệu suất của ứng dụng.

**CHƯƠNG 2 CƠ SỞ LÝ THUYẾT**

## **2.1. Công cụ sử dụng**

### ***2.1.1. Android studio (android app)***

Android Studio là một môi trường phát triển tích hợp (IDE) chính thức của Google, được thiết kế đặc biệt để phát triển ứng dụng di động chạy trên hệ điều hành Android. Đây là công cụ phổ biến và mạnh mẽ được sử dụng bởi các nhà phát triển để tạo ra các ứng dụng đa dạng từ ứng dụng di động cho doanh nghiệp đến các trò chơi giải trí.

Một số điểm nổi bật về Android Studio:

* IDE Chính Thống: Android Studio là một IDE chính thức được Google hỗ trợ, đi kèm với các tính năng tiên tiến và hỗ trợ đa dạng để phát triển ứng dụng Android.
* Công Cụ Phát Triển: Android Studio cung cấp một loạt các công cụ phát triển mạnh mẽ như trình biên dịch, trình gỡ lỗi, trình quản lý mã nguồn, và trình quản lý dự án.
* Tích Hợp Gradle: Gradle là một hệ thống quản lý dự án mạnh mẽ, và Android Studio tích hợp sâu vào Gradle để quản lý các phụ thuộc và quản lý dự án hiệu quả.
* Thiết Kế Giao Diện: Android Studio cung cấp Designer Layout để thiết kế giao diện người dùng của ứng dụng một cách trực quan, kèm theo việc hỗ trợ trực tiếp XML để tùy chỉnh giao diện.
* Mô phỏng và Kiểm Tra: Android Studio đi kèm với Android Emulator cho phép bạn mô phỏng và kiểm tra ứng dụng trên nhiều thiết bị và phiên bản Android khác nhau.
* Hỗ trợ Kotlin: Kotlin là ngôn ngữ lập trình chính thức cho Android và Android Studio cung cấp hỗ trợ toàn diện cho Kotlin, cho phép bạn phát triển ứng dụng bằng cả Java và Kotlin.
* Cộng Đồng Lớn Mạnh: Android Studio có một cộng đồng lớn mạnh, với nhiều tài liệu, hướng dẫn, và cộng đồng hỗ trợ trên Internet.

### ***2.1.2. VS Code (web app)***

Visual Studio Code (VS Code) là một trình biên soạn mã nguồn mở và miễn phí được phát triển bởi Microsoft. Được xây dựng trên nền tảng Electron, VS Code được thiết kế để hỗ trợ nhiều ngôn ngữ lập trình và môi trường phát triển khác nhau.

Một số điểm nổi bật về Visual Studio Code:

* Đa Năng: VS Code là một trình biên soạn mã linh hoạt với hỗ trợ mạnh mẽ cho nhiều ngôn ngữ lập trình như JavaScript, TypeScript, HTML, CSS, C++, Java, Python, và nhiều ngôn ngữ khác nữa.
* Hỗ Trợ Mở Rộng: VS Code cho phép bạn mở rộng chức năng của nó thông qua các tiện ích mở rộng, từ cú pháp màu cho ngôn ngữ đến tích hợp với các công cụ kiểm tra mã nguồn và quản lý phiên bản.
* Hỗ Trợ Git: VS Code tích hợp sâu với Git, cho phép bạn thực hiện các hoạt động như commit, pull, push trực tiếp từ giao diện người dùng.
* Debugging: VS Code cung cấp một bộ công cụ gỡ lỗi mạnh mẽ cho nhiều ngôn ngữ, cho phép bạn gỡ lỗi mã nguồn của mình một cách hiệu quả.
* IntelliSense: VS Code có tính năng IntelliSense thông minh giúp bạn viết mã nhanh chóng và hiệu quả hơn bằng cách cung cấp gợi ý và hoàn thành mã tự động.
* Hỗ Trợ Đa Nền Tảng: VS Code có sẵn trên nhiều nền tảng, bao gồm Windows, macOS và Linux, cho phép bạn làm việc trên bất kỳ máy tính nào mà không bị giới hạn.
* Cộng Đồng Phong Phú: VS Code có một cộng đồng phát triển mạnh mẽ với nhiều tiện ích mở rộng và chủ đề giao diện được tạo ra bởi cộng đồng người dùng.

### ***2.1.3. MongoDB***

MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở được thiết kế để xử lý dữ liệu không cấu trúc và cấu trúc linh hoạt.

Một số điểm nổi bật về MongoDB:

* Cấu trúc dữ liệu linh hoạt: MongoDB lưu trữ dữ liệu dưới dạng JSON-kiểu tài liệu được gọi là BSON (Binary JSON). Điều này cho phép lưu trữ dữ liệu với cấu trúc linh hoạt, cho phép thay đổi cấu trúc dữ liệu một cách dễ dàng và linh hoạt.
* Tính linh động và mở rộng: MongoDB hỗ trợ các cụm và sao lưu dữ liệu tự động, giúp nó mở rộng dễ dàng theo nhu cầu của ứng dụng. Bằng cách sử dụng các cụm dữ liệu và sao lưu phân tán, MongoDB có thể mở rộng để xử lý khối lượng dữ liệu lớn và tải lên nhanh chóng.
* Truy vấn mạnh mẽ: MongoDB cung cấp một ngôn ngữ truy vấn linh hoạt và mạnh mẽ, cho phép truy vấn dữ liệu theo nhiều tiêu chí khác nhau, bao gồm truy vấn theo tài liệu, truy vấn phức tạp và truy vấn theo điều kiện.
* Tích hợp tốt với các ngôn ngữ lập trình: MongoDB cung cấp các API và thư viện cho nhiều ngôn ngữ lập trình phổ biến như Java, Python, JavaScript, Ruby, và nhiều ngôn ngữ khác nữa. Điều này giúp phát triển ứng dụng dễ dàng trên các nền tảng khác nhau.
* Hệ thống tập trung: MongoDB sử dụng mô hình hệ thống tập trung, nơi mỗi bản ghi dữ liệu được lưu trữ trong một tài liệu độc lập. Điều này giúp cải thiện hiệu suất và tăng khả năng mở rộng của hệ thống.
* Dễ dàng quản lý: MongoDB cung cấp các công cụ quản lý và giám sát tích hợp, bao gồm MongoDB Compass và MongoDB Atlas, giúp quản lý dữ liệu và hệ thống một cách hiệu quả.

## **2.3. Ngôn ngữ và công nghệ sử dụng bên trong web app**

### ***2.3.1. NodeJS (web app)***

Node.js là một nền tảng phát triển ứng dụng được xây dựng trên JavaScript runtime của Chrome (V8 Engine).

Một số điểm nổi bật về Node.js:

* JavaScript Everywhere: Node.js cho phép viết mã bằng JavaScript cả ở phía máy chủ và phía máy khách, giúp làm giảm sự phân tách giữa các dự án và tăng cường khả năng tái sử dụng mã.
* Asynchronous và Non-blocking I/O: Một trong những đặc điểm quan trọng của Node.js là sự hỗ trợ mạnh mẽ cho việc xử lý không đồng bộ và I/O không chặn. Điều này giúp Node.js xử lý hàng loạt yêu cầu mà không cần phải chờ đợi đến khi một yêu cầu hoàn thành trước khi bắt đầu yêu cầu khác.
* Mô-đun và Package Ecosystem phong phú: Node.js có một hệ thống mô-đun mạnh mẽ với hàng ngàn các gói mô-đun được cộng đồng phát triển và chia sẻ trên npm (Node Package Manager). Nhờ đó, bạn có thể dễ dàng sử dụng các thư viện và công cụ mạnh mẽ để phát triển ứng dụng.
* Mô-đun hóa và tái sử dụng mã: Node.js khuyến khích việc sử dụng mô-đun và phát triển theo cách mô-đun hóa. Điều này giúp tạo ra mã dễ bảo trì và tái sử dụng, đồng thời cũng thúc đẩy việc phát triển nhanh chóng và linh hoạt.
* Real-time và WebSocket: Node.js được sử dụng rộng rãi trong các ứng dụng real-time như trò chơi trực tuyến, ứng dụng trò chuyện, và ứng dụng cập nhật dữ liệu real-time. Thư viện Socket.IO cung cấp một cách dễ dàng để tạo ra các ứng dụng real-time sử dụng WebSocket.
* Hiệu suất cao: Nhờ kiến trúc non-blocking và asynchronous, Node.js có khả năng xử lý hàng loạt yêu cầu với tài nguyên thấp, đồng thời cung cấp hiệu suất cao.

### ***2.3.2. ReactJS (fontend - web app)***

ReactJS là một thư viện JavaScript mã nguồn mở được sử dụng rộng rãi để xây dựng giao diện người dùng (UI) cho các ứng dụng web.

Một số điểm nổi bật về ReactJS:

* Component-Based: ReactJS được xây dựng trên cơ sở của các thành phần (components), cho phép bạn phân chia giao diện người dùng thành các thành phần nhỏ hơn, độc lập và tái sử dụng. Điều này giúp tăng cường khả năng bảo trì và phát triển ứng dụng.
* Virtual DOM: React sử dụng một cơ chế gọi là Virtual DOM để tối ưu hóa hiệu suất và tăng tốc quá trình render UI. Thay vì cập nhật trực tiếp DOM mỗi khi có thay đổi, React sẽ cập nhật trước Virtual DOM, sau đó so sánh với DOM hiện tại để tìm ra những thay đổi cần được áp dụng, và cuối cùng chỉ cập nhật những phần thay đổi đó.
* One-Way Data Binding: React thực hiện one-way data binding, nghĩa là dữ liệu chỉ được truyền từ component cha xuống các component con. Điều này giúp giảm sự phức tạp của ứng dụng và dễ dàng theo dõi dòng dữ liệu.
* JSX: JSX là một cú pháp cho phép bạn viết HTML trong JavaScript. Điều này giúp làm tăng tính dễ đọc và dễ hiểu của mã, đồng thời cung cấp một cách tiện lợi để tạo và duy trì các thành phần UI.
* Ecosystem và Cộng đồng mạnh mẽ: ReactJS có một hệ sinh thái (ecosystem) phong phú với hàng ngàn các thư viện và công cụ được phát triển bởi cộng đồng lớn. Điều này giúp tăng tốc quá trình phát triển và giảm thời gian triển khai.
* React Native: ReactJS cung cấp React Native, một framework cho phép bạn xây dựng ứng dụng di động bằng JavaScript và React, chia sẻ mã nguồn giữa các nền tảng iOS và Android.

### ***2.3.3. Redux (fontend - web app)***

Redux là một thư viện quản lý trạng thái (state management) cho ứng dụng JavaScript, thường được sử dụng cùng với ReactJS để quản lý trạng thái của ứng dụng web.

Một số điểm nổi bật về Redux:

* Single Source of Truth: Redux giữ cho trạng thái của toàn bộ ứng dụng được lưu trữ trong một "store" duy nhất. Điều này làm cho việc quản lý trạng thái trở nên đơn giản hơn và dễ dàng theo dõi.
* State is Read-Only: Trạng thái trong Redux là read-only, điều này có nghĩa là bạn không thể thay đổi trạng thái trực tiếp. Thay vào đó, bạn cần phải tạo ra các "actions" để mô tả các sự kiện (events) hoặc thay đổi trạng thái, và sử dụng các "reducers" để xử lý các hành động đó và cập nhật trạng thái.
* Predictable State Changes with Reducers: Reducers là các hàm pure functions nhận vào một trạng thái hiện tại và một action, và trả về một trạng thái mới. Bằng cách này, Redux đảm bảo rằng các thay đổi trạng thái của ứng dụng là có thể dự đoán và nhất quán.
* Unidirectional Data Flow: Redux thực hiện một luồng dữ liệu một chiều, nghĩa là dữ liệu chỉ di chuyển theo một hướng từ "actions" đến "reducers" và từ "reducers" đến "store". Điều này giúp giảm bớt sự phức tạp của ứng dụng và dễ dàng theo dõi dòng dữ liệu.
* Middleware: Redux cho phép sử dụng middleware để mở rộng và tùy chỉnh quá trình xử lý actions trước khi chúng đến reducers. Middleware có thể được sử dụng để thực hiện các chức năng như ghi nhật ký, xử lý bất đồng bộ, hoặc thậm chí là thực hiện logic phức tạp cho actions.
* DevTools Extension: Redux cung cấp các công cụ mở rộng cho trình duyệt (như Redux DevTools Extension) giúp theo dõi và debug ứng dụng Redux một cách hiệu quả.

### ***2.3.4. Express (backend – web app)***

Express là một framework web backend cho Node.js, được sử dụng để xây dựng các ứng dụng web và API một cách dễ dàng và nhanh chóng.

Một số điểm nổi bật về Express:

* Minimal và Unopinionated: Express được thiết kế để có cấu trúc tối giản và linh hoạt, cho phép nhà phát triển tự do lựa chọn cách tổ chức dự án của mình. Nó không áp đặt quá nhiều quy tắc và cách tiếp cận, giúp tạo ra mã nguồn linh hoạt và dễ bảo trì.
* Routing: Express cung cấp một cách tiếp cận linh hoạt để xử lý các yêu cầu HTTP thông qua việc định nghĩa các tuyến đường (routes). Bằng cách này, bạn có thể định nghĩa các điều hướng và xử lý các yêu cầu GET, POST, PUT, DELETE, vv một cách dễ dàng.
* Middleware: Middleware là một tính năng quan trọng của Express, cho phép bạn thực hiện các chức năng như xác thực, ghi nhật ký, xử lý ngoại lệ, và nhiều hơn nữa trước khi yêu cầu đến các xử lý tiếp theo. Điều này giúp tạo ra mã nguồn modul và dễ mở rộng.
* Template Engine Support: Express hỗ trợ nhiều động cơ mẫu (template engine) như Pug, EJS, và Handlebars để tạo ra nội dung HTML động. Điều này giúp dễ dàng tạo ra các trang web động và linh hoạt trong việc hiển thị dữ liệu.
* Error Handling: Express cung cấp cơ chế xử lý ngoại lệ mạnh mẽ cho các ứng dụng, cho phép bạn xử lý các ngoại lệ một cách linh hoạt và kiểm soát luồng của ứng dụng trong trường hợp xảy ra lỗi.
* Integration with Node.js: Express là một framework tinh gọn và xây dựng trên nền tảng Node.js, điều này cho phép bạn sử dụng tất cả các tính năng và thư viện của Node.js trong ứng dụng Express của bạn.

***2.3.5. Vite***

Vite là một công cụ phát triển web hiệu quả được tạo ra bởi Evan You, người cùng sáng lập Vue.js. Được thiết kế để cung cấp một quy trình phát triển nhanh chóng và mượt mà, Vite sử dụng các công nghệ mới như ES Modules và HTTP/2 để mang lại trải nghiệm phát triển tốt nhất cho các ứng dụng web hiện đại.

Một trong những điểm đặc biệt của Vite là cách nó xử lý các tệp nguồn. Thay vì phải biên dịch toàn bộ ứng dụng trước khi chạy, Vite chỉ biên dịch các tệp cụ thể khi chúng được yêu cầu. Điều này giúp giảm thời gian khởi động và tải lại trình duyệt, tăng tốc độ phát triển và thử nghiệm.

Vite hỗ trợ nhiều framework frontend như Vue.js, React và Svelte, và tích hợp các tính năng như Hot Module Replacement (HMR) để tự động làm mới trình duyệt khi có thay đổi trong mã nguồn. Điều này giúp các nhà phát triển dễ dàng theo dõi và thử nghiệm các thay đổi mà không cần phải làm mới trang hoặc khởi động lại ứng dụng.

## **2.4. Ngôn ngữ và công nghệ sử dụng bên trong android app**

### ***2.4.1. Java***

Java là một ngôn ngữ lập trình phổ biến và mạnh mẽ được phát triển bởi Sun Microsystems (sau này được mua lại bởi Oracle).

Một số điểm nổi bật về Java:

* Đa Nền Tảng: Một trong những ưu điểm lớn nhất của Java là khả năng chạy trên nhiều nền tảng khác nhau mà không cần biên dịch lại. Điều này làm cho Java trở thành một lựa chọn lý tưởng cho việc phát triển các ứng dụng đa nền tảng và dễ dàng triển khai trên môi trường khác nhau.
* OOP (Object-Oriented Programming): Java được thiết kế dựa trên một mô hình lập trình hướng đối tượng mạnh mẽ. Nó hỗ trợ các khái niệm quan trọng như lớp, đối tượng, kế thừa, đóng gói và đa hình, giúp tạo ra mã nguồn cấu trúc hợp lý và dễ bảo trì.
* Bảo Mật: Java có một môi trường thực thi kiểu sandbox được thiết kế để ngăn chặn các vấn đề bảo mật như virus và phần mềm độc hại. Điều này làm cho Java trở thành một lựa chọn phù hợp cho việc phát triển các ứng dụng web và ứng dụng nơi mà an ninh là ưu tiên hàng đầu.
* Khả Năng Mở Rộng: Java cung cấp một loạt các thư viện và framework phong phú để phát triển các ứng dụng, từ ứng dụng desktop đến ứng dụng web và di động. Các công cụ như Spring Framework, Hibernate và JavaFX giúp tăng cường khả năng phát triển và mở rộng của ứng dụng Java.
* Hiệu Suất: Mặc dù Java thường được đánh giá là chậm hơn so với một số ngôn ngữ lập trình khác như C ++, nhưng nó vẫn cung cấp hiệu suất đủ tốt cho hầu hết các ứng dụng. Các công cụ như Just-In-Time (JIT) compiler giúp tối ưu hóa mã nguồn để cải thiện hiệu suất thời gian chạy.
* Cộng Đồng Mạnh Mẽ: Java có một cộng đồng lớn và tích cực, với hàng ngàn người dùng, nhà phát triển và công ty sử dụng và đóng góp vào việc phát triển ngôn ngữ. Điều này giúp đảm bảo rằng Java luôn được cập nhật và phát triển với các công nghệ mới nhất.

### ***2.4.2. Kotlin***

Kotlin là một ngôn ngữ lập trình hiện đại, đa nền tảng, được phát triển bởi JetBrains.

Một số điểm nổi bật về Kotlin:

* Tích Hợp Tốt với Java: Kotlin được thiết kế để tương thích tốt với mã nguồn Java hiện có. Điều này có nghĩa là bạn có thể tích hợp mã nguồn Kotlin vào dự án Java hiện có mà không gặp phải vấn đề lớn.
* An Toàn Với Null: Kotlin giảm thiểu nguy cơ lỗi Null Pointer Exception (NPE) bằng cách tích hợp kiểm tra null vào ngôn ngữ. Bạn có thể xác định một biến có thể chứa giá trị null hoặc không thể, giúp giảm thiểu các lỗi runtime liên quan đến null.
* Lập Trình Hướng Chức Năng và Hướng Đối Tượng: Kotlin hỗ trợ cả lập trình hướng đối tượng (OOP) và lập trình hàm (FP). Điều này cho phép bạn sử dụng các tính năng của cả hai phong cách lập trình để phát triển ứng dụng.
* Tính Đa Nền Tảng (Multiplatform): Kotlin cung cấp tính năng đa nền tảng, cho phép bạn chia sẻ logic kinh doanh giữa các nền tảng khác nhau như Android, iOS và máy chủ. Điều này giúp tăng tốc quá trình phát triển và duy trì mã nguồn.
* Cú Pháp Ngắn Gọn và Đọc Hiểu: Kotlin có cú pháp ngắn gọn và dễ đọc hơn so với Java. Điều này giúp làm giảm số lượng mã cần phải viết và giúp tăng hiệu suất phát triển.
* Hỗ Trợ Tính Toán và Xử Lý Dữ Liệu: Kotlin cung cấp nhiều tính năng hữu ích như extension functions, data classes và lambdas, giúp tăng cường khả năng xử lý dữ liệu và làm cho mã nguồn của bạn trở nên ngắn gọn và dễ hiểu hơn.

### ***2.4.3. Retrofit và thư viện Gson***

Retrofit và thư viện Gson là hai công cụ phổ biến được sử dụng trong phát triển ứng dụng Android để làm việc với dữ liệu từ các API web.

Retrofit:

* HTTP Client: Retrofit là một HTTP client library cho Android và Java, được sử dụng để gửi và nhận các yêu cầu HTTP và xử lý các phản hồi tương ứng từ máy chủ.
* Annotation-based API Declaration: Retrofit cho phép bạn định nghĩa API dưới dạng các phương thức gọi trực tiếp từ giao diện API, sử dụng các chú thích như @GET, @POST, @PUT, @DELETE v.v.
* Integrates with Gson or other Converters: Retrofit có thể tích hợp với các thư viện chuyển đổi (converters) như Gson để tự động chuyển đổi dữ liệu JSON thành các đối tượng Java và ngược lại.
* Asynchronous and Synchronous Requests: Retrofit hỗ trợ cả yêu cầu bất đồng bộ và đồng bộ. Bạn có thể thực hiện các yêu cầu mạng mà không cần phải lo lắng về việc chặn luồng chính.
* Easy Error Handling: Retrofit cung cấp cơ chế xử lý lỗi dễ dàng thông qua callback hoặc sử dụng kiểu dữ liệu ‘Response<T>’.

Gson:

* JSON Serialization and Deserialization: Gson là một thư viện Java dùng để chuyển đổi các đối tượng Java thành dạng JSON (serialization) và ngược lại (deserialization).
* Customizable Conversion: Gson cho phép bạn tùy chỉnh cách chuyển đổi các đối tượng Java thành JSON bằng cách sử dụng các chú thích và quy tắc.
* Supports Complex Data Structures: Gson có thể xử lý các cấu trúc dữ liệu phức tạp như danh sách, mảng, lồng nhau và các đối tượng nhúng.
* Performance Optimizations: Gson được tối ưu hóa để có hiệu suất cao và tiết kiệm tài nguyên, giúp giảm thiểu thời gian và bộ nhớ cần thiết để thực hiện các thao tác chuyển đổi.
* Integrates Seamlessly with Retrofit: Gson là một trong những converter phổ biến nhất được sử dụng với Retrofit để xử lý dữ liệu JSON từ các phản hồi từ máy chủ.

### ***2.4.4. Cơ sở dữ liệu cục bộ SQLite***

Cơ sở dữ liệu cục bộ SQLite là một phần quan trọng trong phát triển ứng dụng di động Android, cho phép lưu trữ và quản lý dữ liệu trên thiết bị của người dùng.

Một số điểm nổi bật về SQLite trong Android:

* Embedded Database: SQLite là một cơ sở dữ liệu nhúng, được tích hợp trực tiếp vào ứng dụng Android mà không cần phải cài đặt riêng biệt.
* Lightweight and Efficient: SQLite là một cơ sở dữ liệu nhẹ, được thiết kế để tiêu tốn ít tài nguyên hệ thống và hoạt động một cách hiệu quả trên các thiết bị di động.
* SQL-Compatible: SQLite sử dụng ngôn ngữ truy vấn cấu trúc (SQL) cho các thao tác đọc và ghi dữ liệu, giúp dễ dàng thực hiện các thao tác như tìm kiếm, sắp xếp và lọc dữ liệu.
* Transactional Support: SQLite hỗ trợ giao dịch ACID (Atomicity, Consistency, Isolation, Durability), cho phép bạn thực hiện các hoạt động ghi dữ liệu một cách an toàn và đảm bảo tính nhất quán của dữ liệu.
* Database Helper Class: Trong Android, bạn có thể sử dụng lớp SQLiteOpenHelper để quản lý cơ sở dữ liệu SQLite. Lớp này cung cấp các phương thức để tạo, nâng cấp và thực hiện các thao tác với cơ sở dữ liệu.
* Content Provider: Content Provider là một thành phần Android cho phép bạn chia sẻ dữ liệu giữa các ứng dụng. Bạn có thể sử dụng Content Provider để cung cấp quyền truy cập vào dữ liệu từ cơ sở dữ liệu SQLite của ứng dụng cho các ứng dụng khác.
* Secure: SQLite hỗ trợ mã hóa dữ liệu và cơ chế bảo mật khác nhau để bảo vệ dữ liệu trước các cuộc tấn công.
* Backup và Restore: Bạn có thể dễ dàng sao lưu và khôi phục dữ liệu từ cơ sở dữ liệu SQLite trên thiết bị Android.

## **2.5. Thiết kế cơ sở dữ liệu**

### ***2.5.1. Mô hình dữ liệu cho sản phẩm, người dùng và hóa đơn***

Bảng Sản phẩm (Products):

* product\_id: Khóa chính, định danh duy nhất của sản phẩm.
* name: Tên sản phẩm.
* description: Mô tả về sản phẩm.
* price: Giá sản phẩm.
* image: Đường dẫn tới hình ảnh của sản phẩm.
* manufacturer\_id: Khóa ngoại, tham chiếu đến nhà sản xuất của sản phẩm.

Bảng Nhà sản xuất (Manufacturers):

* manufacturer\_id: Khóa chính, định danh duy nhất của nhà sản xuất.
* name: Tên của nhà sản xuất.

Bảng Người dùng (Users):

* user\_id: Khóa chính, định danh duy nhất của người dùng.
* email: Địa chỉ email của người dùng (có thể là tên đăng nhập).
* password: Mật khẩu của người dùng (được băm hoặc mã hóa).
* fullname: Họ và tên của người dùng.
* address: Địa chỉ của người dùng.
* phone: Số điện thoại của người dùng.

Bảng Hóa đơn (Bills):

* bill\_id: Khóa chính, định danh duy nhất của hóa đơn.
* user\_id: Khóa ngoại, tham chiếu đến người dùng tạo hóa đơn.
* total\_price: Tổng giá trị của hóa đơn.
* sale\_date: Ngày tạo hóa đơn.

### ***2.5.2. Thiết kế cấu trúc cơ sở dữ liệu cho ứng dụng***

Bảng Sản phẩm (Products):

* product\_id (INTEGER, PRIMARY KEY)
* name (TEXT)
* description (TEXT)
* price (INTEGER)
* image (TEXT)
* manufacturer\_id(INTEGER,FOREIGNKEYREFERENCES Manufacturers(manufacturer\_id))

Bảng Nhà sản xuất (Manufacturers):

* manufacturer\_id (INTEGER, PRIMARY KEY)
* name (TEXT)

Bảng Người dùng (Users):

* user\_id (INTEGER, PRIMARY KEY)
* email (TEXT, UNIQUE)
* password (TEXT)
* fullname (TEXT)
* address (TEXT)
* phone (TEXT)

Bảng Hóa đơn (Bills):

* bill\_id (INTEGER, PRIMARY KEY)
* user\_id (INTEGER, FOREIGN KEY REFERENCES Users(user\_id))
* total\_price (INTEGER)
* sale\_date (TEXT)

# **CHƯƠNG 3 THIẾT KẾ VÀ TRIỂN KHAI**

## **3.1. Sơ đồ use case**

**A screenshot of a computer

Description automatically generated**

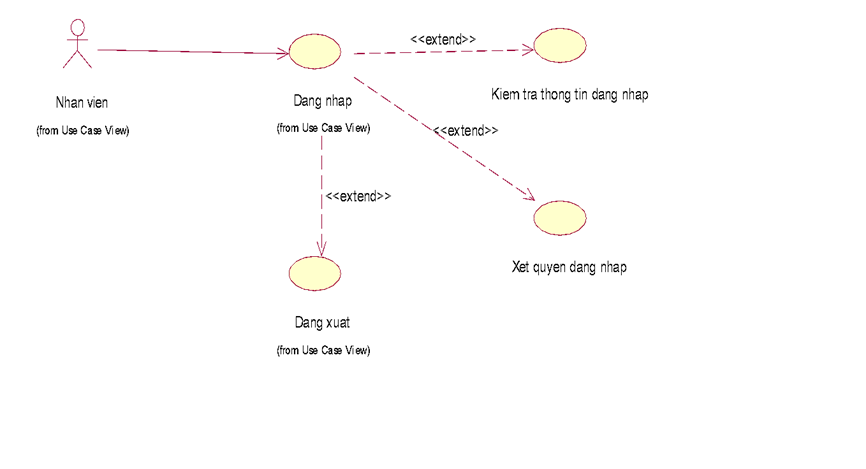
*Hình 3.1 Sơ đồ use case tổng quát*

### ***3.1.1. Xác định use case***

Dựa vào yêu cầu bài toán, ta có các actor sau: Quản trị viên (Admin), người xem (Viewer) và Khách hàng (Customer)

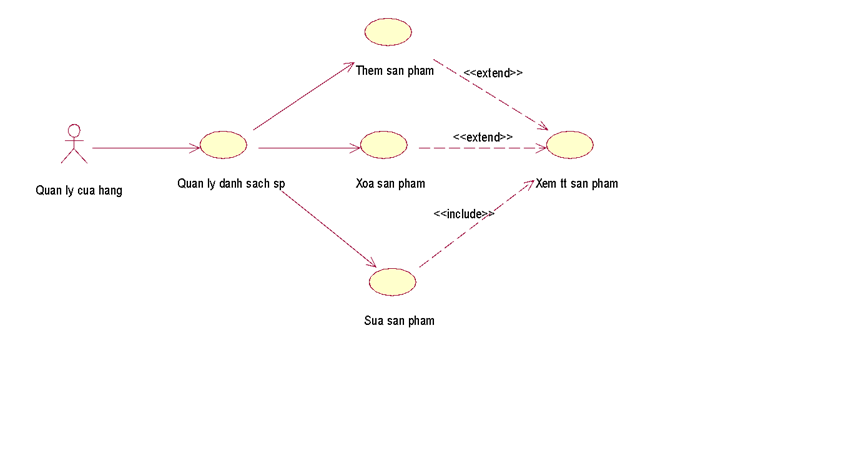
|  |  |
| --- | --- |
| **Actor** | **Use Case** |
| Quản trị viên (Admin) | Đăng nhập, quản lý thông tin sản phẩm, quản lý loại sản phẩm, quản lý đơn hàng. |
| Người xem (Viewer) | Đăng ký tài khoản, xem thông tin sản phẩm, tìm kiếm sản phẩm, quản lý giỏ hàng |
| Khách hàng (Customer) | Đăng nhập, xem thông tin sản phẩm, tìm kiếm sản phẩm, quản lý giỏ hàng, thanh toán. |

### ***3.1.2. Biểu đồ Use Case đăng nhập, đăng xuất***



*Hình 3.2 Sơ đồ Use Case đăng nhập, đăng xuất*

### ***3.1.3. Biểu đồ Use Case Quản lý sản phẩm***



*Hình 3.3. Sơ đồ Use Case quản lý danh mục sản phẩm*

### ***3.1.4. Biểu đồ Use Case Case quản lý hóa đơn***

***A diagram of a diagram

Description automatically generated***

*Hình 3.4 Sơ đồ Use Case quản lý hóa đơn*

## **3.2. Phân tích yêu cầu**

Chức năng của ứng dụng: các tính năng cần có của ứng dụng, bao gồm quản lý sản phẩm, giỏ hàng, đăng nhập, đăng ký, thanh toán, và xem hóa đơn.

Yêu cầu về dữ liệu: Xác định loại dữ liệu cần thu thập và lưu trữ, bao gồm thông tin về sản phẩm, người dùng, giỏ hàng, và hóa đơn.

Giao diện người dùng: yêu cầu về giao diện người dùng, bao gồm các màn hình, nút bấm, trường nhập liệu, và cách tương tác giữa người dùng và ứng dụng.

Hiệu suất và bảo mật: Xác định các yêu cầu về hiệu suất của hệ thống, bao gồm thời gian phản hồi, tải trọng dữ liệu, và bảo mật thông tin cá nhân của người dùng.

Phân quyền truy cập: Đặt ra các yêu cầu về phân quyền truy cập để đảm bảo chỉ những người được phép mới có thể truy cập và thực hiện các chức năng cụ thể trong ứng dụng.

### ***3.2.1. Yêu cầu chức năng***

Quản lý sản phẩm:

* Hiển thị danh sách sản phẩm.
* Tìm kiếm sản phẩm theo tên, nhà sản xuất, hoặc danh mục.
* Xem chi tiết sản phẩm với mô tả, hình ảnh, giá cả, và thông số kỹ thuật.
* Thêm sản phẩm vào giỏ hàng từ trang chi tiết sản phẩm.

Quản lý giỏ hàng:

* Hiển thị giỏ hàng với danh sách sản phẩm và tổng giá.
* Thay đổi số lượng sản phẩm trong giỏ hàng hoặc xóa sản phẩm khỏi giỏ hàng.
* Tính tổng giá tiền của các sản phẩm trong giỏ hàng.

Quản lý đơn hàng:

* Tạo đơn hàng mới sau khi thanh toán thành công.
* Hiển thị lịch sử đơn hàng của người dùng.
* Xem chi tiết đơn hàng bao gồm thông tin về sản phẩm, số lượng, và tổng giá.

Đăng nhập và đăng ký:

* Cho phép người dùng đăng nhập vào tài khoản hiện có hoặc đăng ký tài khoản mới.
* Xác thực thông tin người dùng và bảo mật mật khẩu.

### ***3.2.2. Yêu cầu phi chức năng***

Hiệu suất:

* Ứng dụng phải có thời gian phản hồi nhanh chóng, giảm thiểu thời gian chờ đợi của người dùng.
* Xử lý dữ liệu một cách hiệu quả để tránh trễ hoặc giảm bị treo ứng dụng.

Bảo mật:

* Bảo vệ thông tin cá nhân của người dùng bằng cách mã hóa dữ liệu và cung cấp cơ chế xác thực an toàn.
* Ngăn chặn các cuộc tấn công từ bên ngoài bằng cách triển khai các biện pháp bảo mật phù hợp.

Khả năng mở rộng:

* Thiết kế ứng dụng để dễ dàng mở rộng và thích ứng với việc thêm tính năng hoặc cải thiện sau này.
* Sử dụng kiến trúc linh hoạt và các công nghệ có thể mở rộng để đáp ứng nhu cầu tăng trưởng của ứng dụng.

### ***3.2.3. Phân tích nhu cầu của người dùng***

Nhu cầu sản phẩm:

* Người dùng muốn có quy trình mua sắm thuận tiện và nhanh chóng cho việc mua điện thoại di động.
* Họ muốn có thể tìm kiếm, so sánh và xem các sản phẩm từ nhiều nhà sản xuất và thương hiệu khác nhau.

Nhu cầu giao diện người dùng

* Người dùng mong đợi một giao diện người dùng thân thiện, dễ sử dụng và dễ điều hướng.
* Họ muốn có trải nghiệm mua sắm trực tuyến mượt mà và thú vị trên các thiết bị di động của mình.

Nhu cầu bảo mật và an toàn:

* Người dùng quan tâm đến việc bảo vệ thông tin cá nhân và tài khoản của họ khi thực hiện thanh toán trực tuyến.
* Họ mong đợi ứng dụng cung cấp các biện pháp bảo mật để ngăn chặn việc truy cập trái phép vào thông tin cá nhân hoặc tài khoản của họ.

## **3.3. Thiết kế giao diện người dùng**

### ***3.3.1. Trang đăng nhập***

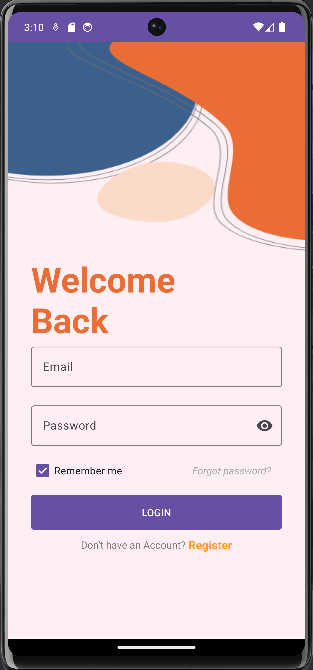
Web app

A screenshot of a login box

Description automatically generated

*Hình 3.5 Trang đăng nhập (web app)*

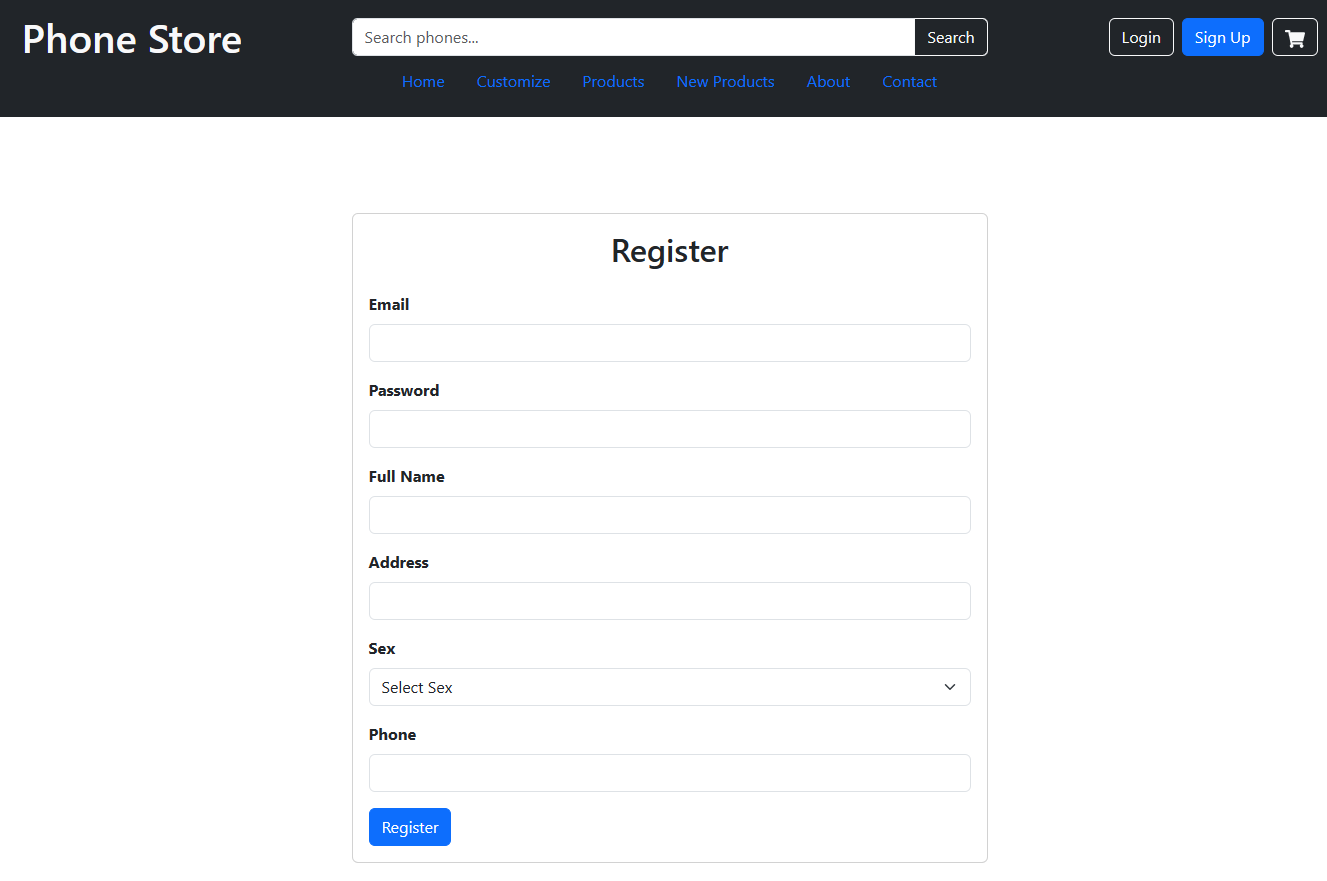
Android app



*Hình 3.6 Trang đăng nhập (android app)*

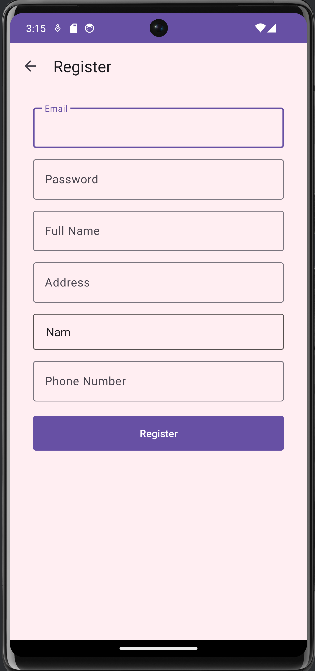
### ***3.3.2. Trang đăng ký***

Web app



*Hình 3.7 Trang đăng kí (web app)*

Android app



*Hình 3.8 Trang đăng ký (web app)*

### ***3.3.3. Trang chủ***

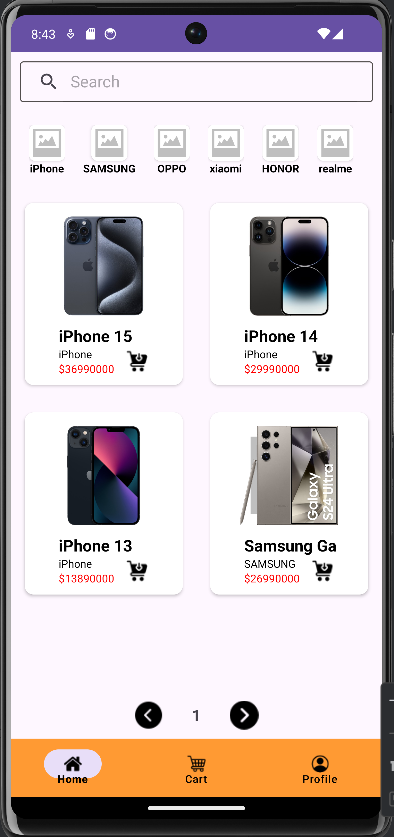
Web app

A screenshot of a phone

Description automatically generated

*Hình 3.9 Trang chủ (web app)*

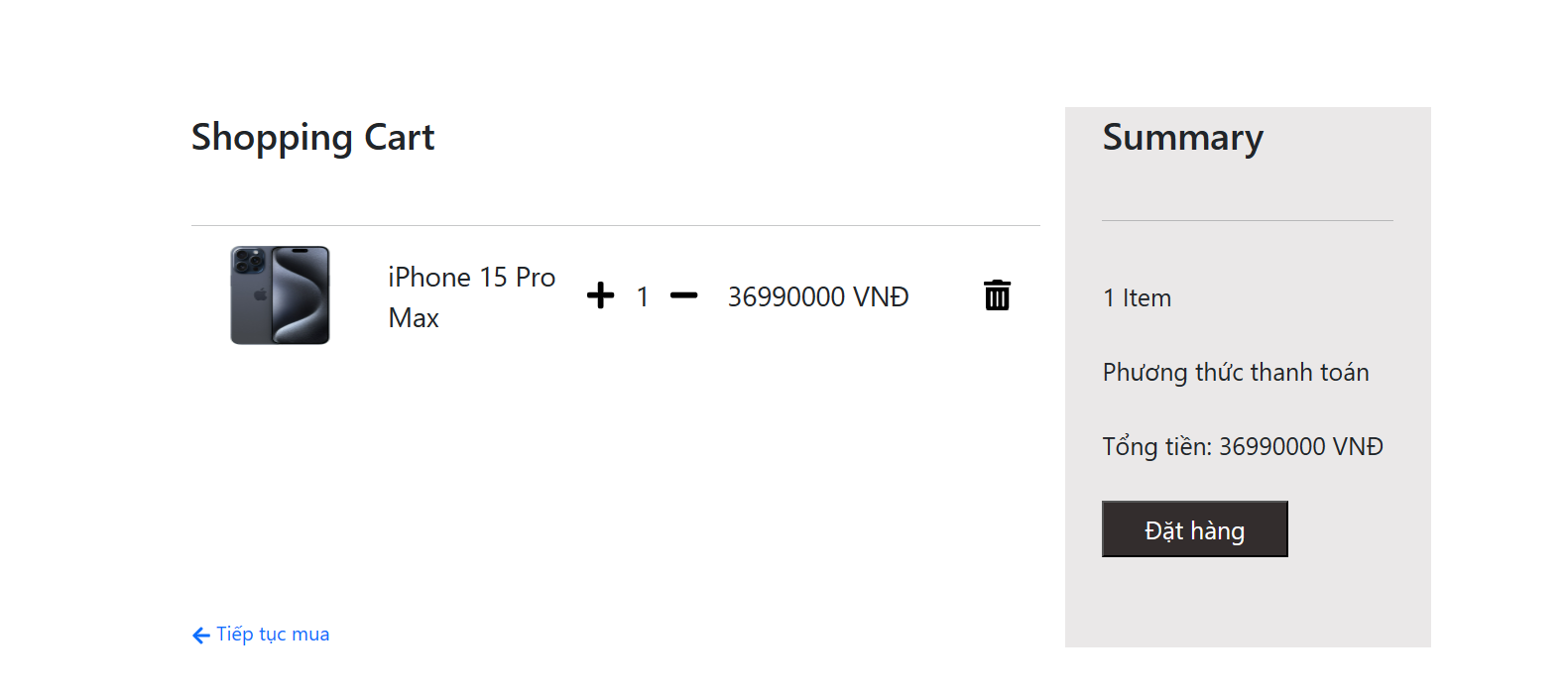
Android app



*Hình 3.10 Trang chủ (android app)*

### ***3.3.4. Giỏ hàng***

Web app



*Hình 3.11 Trang giỏ hàng (web app)*

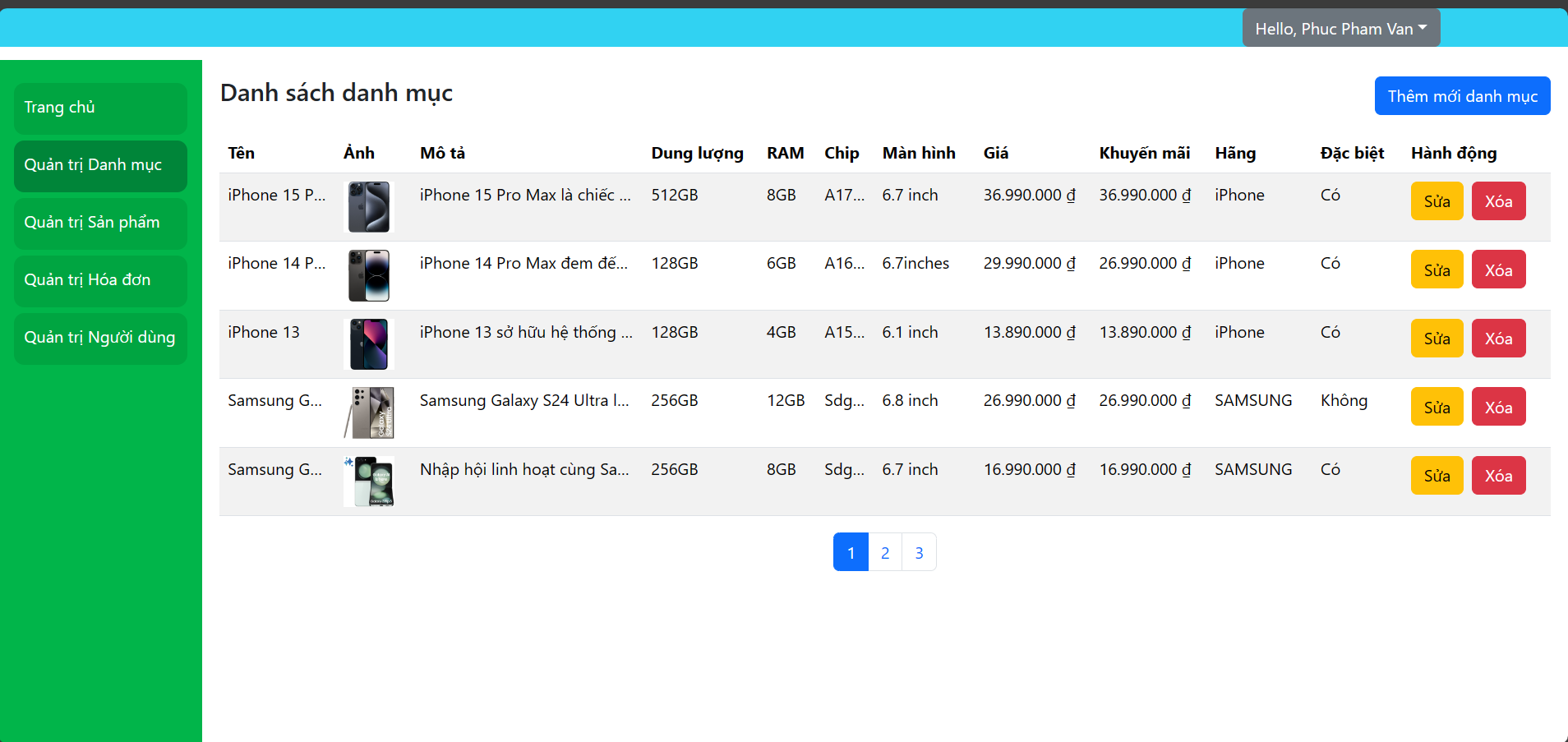
Android app

A screen shot of a phone

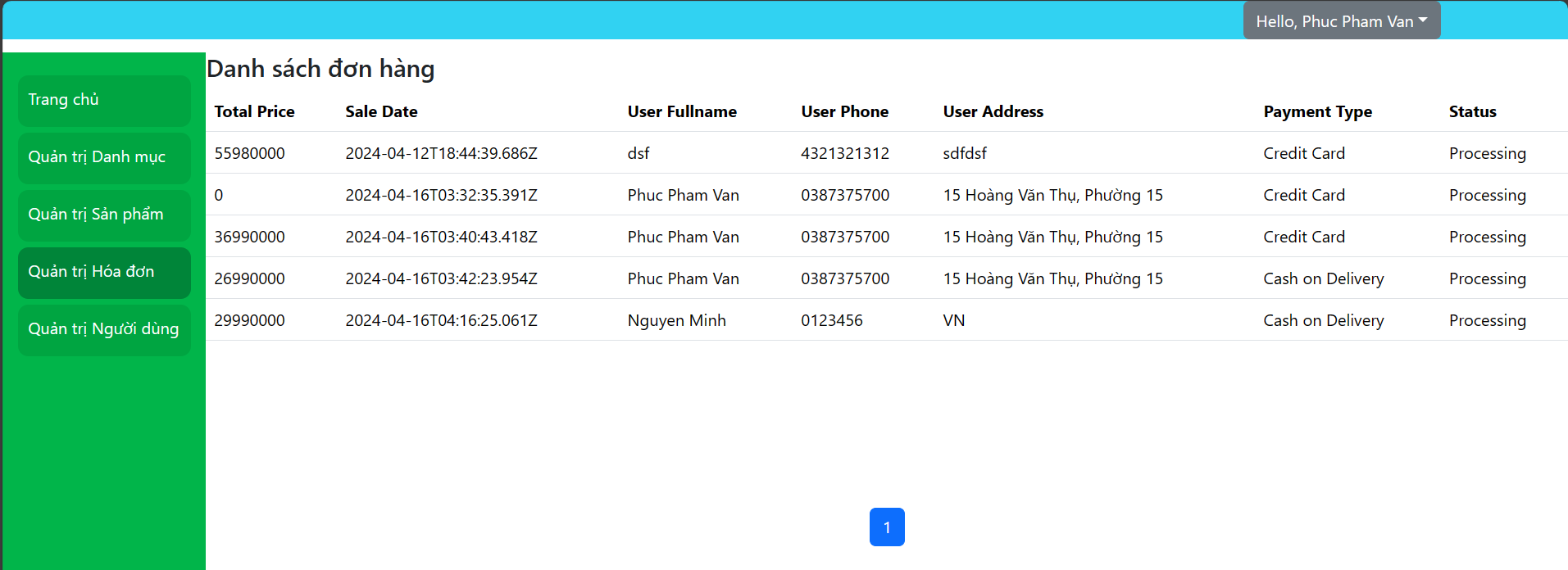
Description automatically generated

*Hình 3.12 Trang giỏ hàng (android app)*

### ***3.3.5. Trang quản trị thêm, xoá, sửa sản phẩm, hoá đơn mua hàng.***

******

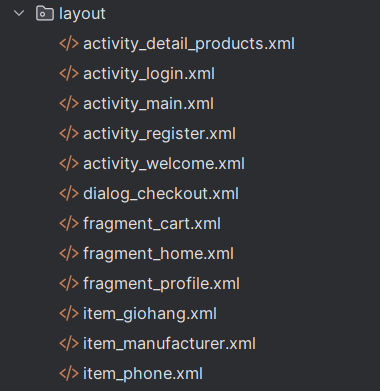
*Hình 3.13 Trang quản trị thêm,xoá, sửa sản phẩm*



*Hình 3.14 Trang quản trị hoá đơn*

## **3.4. Hiện Thực Ứng Dụng (Android app)**

### ***3.4.1. Thiết kế giao diện người dùng***



*Hình 3.15 Mô tả layout (android app)*

Tạo các layout XML cho màn hình đăng nhập, đăng ký, trang chủ, giỏ hàng, ...

Bao gồm các trường nhập liệu như email, mật khẩu, xác nhận mật khẩu, tên đầy đủ, địa chỉ, và số điện thoại.

Thêm các button đăng nhập đăng ký, xoá khỏi giỏ hàng, thêm vào giỏ hàng, các nút phía navbar (trang chủ, trang giỏ hàng, trang thông tin cá nhân)

### ***3.4.2. Xử lý logic, quản lý trạng thái đăng nhập***

Xử lý các logic:

* Sử dụng Retrofit để tương tác với API backend.
* Xử lý các sự kiện khi người dùng nhấn nút.
* Kiểm tra dữ liệu đầu vào từ người dùng và gửi yêu cầu tương ứng đến server.
* Xử lý phản hồi từ server sau khi thực hiện đăng nhập hoặc đăng ký.

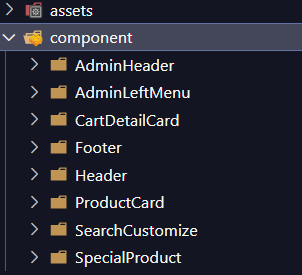
Quản lý trạng thái đăng nhập:

* Lưu trữ thông tin người dùng đã đăng nhập (ví dụ: ID, tên người dùng) vào SharedPreferences hoặc lưu trữ cục bộ.
* Sử dụng thông tin người dùng đã đăng nhập để cung cấp các tính năng được yêu cầu như xem hồ sơ cá nhân hoặc thực hiện các thao tác liên quan đến tài khoản.

## **3.5. Hiện Thực Ứng Dụng (Web app)**

### ***3.5.1. Thiết kế giao diện người dùng ( fontend)***

Sử dụng các framework frontend như React, Vue.js và Svelte để xây dựng giao diện người dùng. Các thành phần giao diện như buttons, forms, và hiển thị danh sách sản phẩm sẽ được tạo trong các file component riêng biệt.



*Hình 3.12 Mô tả layout (web app)*

Sử dụng CSS hoặc SCSS để tạo giao diện đẹp và phản ánh nhãn hiệu của ứng dụng. Các file CSS/SCSS sẽ được sử dụng để tùy chỉnh giao diện của các component.

### ***3.5.2. Xử lý logic (backend)***

Sử dụng Node.js và Express để xây dựng phần backend của ứng dụng. Các route được định nghĩa trong các file route sẽ xử lý các yêu cầu từ frontend và tương tác với cơ sở dữ liệu MongoDB.

Sử dụng MongoDB để lưu trữ dữ liệu của ứng dụng như sản phẩm, người dùng và hóa đơn. Các model schema được định nghĩa trong các file schema sẽ ánh xạ dữ liệu từ cơ sở dữ liệu vào các đối tượng JavaScript để xử lý.

# **CHƯƠNG 4 KẾT LUẬN**

## **5.1. Tóm tắt kết quả**

Lựa chọn đề tài: Tạo một ứng dụng web bán điện thoại di động.

Mục tiêu nghiên cứu: Xây dựng một hệ thống hoàn chỉnh cho việc quản lý sản phẩm, danh mục, người dùng và hóa đơn.

Giới thiệu về Node.js, Express, React, Redux, MongoDB và các công nghệ liên quan khác ở nền tảng web app.

Giới thiệu về Java, Retrofit, Gson, SQlite ở nền tảng android app.

Việc phân tích yêu cầu và nhu cầu của người dùng đã giúp xây dựng một ứng dụng có tính năng đa dạng và phù hợp. Thiết kế cơ sở dữ liệu được xây dựng có cấu trúc và mô hình rõ ràng, giúp quản lý dữ liệu một cách hiệu quả.

Kết quả của tiểu luận là một ứng dụng linh hoạt, dễ sử dụng và có khả năng mở rộng. Việc triển khai ứng dụng đã được chuẩn bị kỹ lưỡng và được thực hiện một cách thành công. Điều này đảm bảo rằng ứng dụng có thể đáp ứng nhu cầu của người dùng một cách tốt nhất và mang lại trải nghiệm tuyệt vời nhất.

## **5.2. Đánh Giá và Hướng Phát Triển**

Đánh Giá:

* Ứng dụng đã hoàn thành các chức năng chính như đăng nhập, đăng ký, xem danh sách sản phẩm, thêm vào giỏ hàng, và thanh toán hóa đơn.
* Giao diện người dùng được thiết kế một cách hợp lý, dễ sử dụng và thân thiện với người dùng.
* Cơ sở dữ liệu được tổ chức cẩn thận và có cấu trúc, giúp quản lý dữ liệu hiệu quả.

Hướng Phát Triển:

* Tối Ưu Hóa Hiệu Năng: Tiếp tục tối ưu hóa hiệu năng của ứng dụng, đặc biệt là trong việc truy vấn cơ sở dữ liệu và tải dữ liệu từ máy chủ.
* Phát Triển Tính Năng Mở Rộng: Bổ sung các tính năng mới như đánh giá sản phẩm, tìm kiếm sản phẩm, và tùy chỉnh hóa đơn.
* Phát Triển Phiên Bản Đa Nền Tảng: Xây dựng phiên bản cho các nền tảng khác như iOS để mở rộng phạm vi sử dụng của ứng dụng.
* Tăng Cường Bảo Mật: Cải thiện lớp bảo mật của ứng dụng để đảm bảo an toàn thông tin cá nhân của người dùng.
* Tích Hợp Thanh Toán: Bổ sung các phương thức thanh toán mới và tích hợp các cổng thanh toán phổ biến để tăng tính tiện lợi cho người dùng.
* Nâng Cấp Giao Diện Người Dùng: Cải thiện giao diện người dùng để tạo ra trải nghiệm người dùng tốt hơn và thú vị hơn.

## **5.3. Kết luận**

Trong suốt quá trình nghiên cứu và phát triển ứng dụng bán điện thoại di động, em đã thực hiện một hành trình hết sức hứng thú và hữu ích. Điểm khởi đầu là việc chọn đề tài, một quyết định quan trọng đòi hỏi sự kỳ công và cân nhắc để đảm bảo rằng đề tài có sự hấp dẫn và tiềm năng phát triển. Sau đó, quá trình phân tích yêu cầu và thiết kế cơ sở dữ liệu đã đặt nền móng vững chắc cho việc phát triển ứng dụng.

Khi bước vào giai đoạn triển khai, em đã gặp phải nhiều thách thức và cần phải áp dụng kiến thức lý thuyết vào thực tế. Việc hiện thực các chức năng như đăng nhập, đăng ký, quản lý sản phẩm và quản lý giỏ hàng đòi hỏi sự tỉ mỉ và kiên nhẫn. Nhưng nhìn lại, mỗi khó khăn đã trở thành cơ hội để học hỏi và phát triển kỹ năng.

Quan trọng hơn, việc thực hiện ứng dụng từ đầu đến cuối đã giúp em hiểu rõ hơn về các công nghệ và công cụ như Android Studio, Retrofit, MongoDB và ExpressJS. Bằng cách thử nghiệm và áp dụng kiến thức này vào thực tế, chúng ta đã có cơ hội làm quen với quy trình phát triển phần mềm và rèn luyện kỹ năng lập trình.

Trong quá trình đánh giá và đề xuất hướng phát triển tiếp theo, em nhận ra rằng việc tiếp tục tối ưu hóa ứng dụng, mở rộng tính năng và tăng cường bảo mật sẽ là những điểm chính cần tập trung. Đồng thời, việc phát triển phiên bản đa nền tảng và tích hợp các phương thức thanh toán mới cũng sẽ là những bước quan trọng trong việc nâng cao trải nghiệm người dùng và mở rộng phạm vi hoạt động của ứng dụng.

Tiểu luận này không chỉ là một cuộc hành trình nghiên cứu và phát triển ứng dụng, mà còn là một cơ hội để khám phá, học hỏi và trưởng thành trong lĩnh vực công nghệ thông tin.

# **TÀI LIỆU THAM KHẢO**

1. Android Studio 4.0 Development Essentials - Java Edition, Tác giả: Neil Smyth, Nhà xuất bản: Techotopia, Xuất bản năm: 2020.
2. Node.js Design Patterns - Second Edition, Tác giả: Mario Casciaro, Luciano Mammino, Nhà xuất bản: Packt Publishing, Xuất bản năm: 2018.
3. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux, Tác giả: Kirupa Chinnathambi, Nhà xuất bản: Addison-Wesley Professional, Xuất bản năm: 2018.
4. Thiết kế cơ sở dữ liệu SQL và NoSQL hiện đại, Tác giả: Võ Đức Thoại, Nhà xuất bản: Thống kê, Xuất bản năm: 2021.
5. Học Reactjs Cơ bản đến Nâng cao, Tác giả: Vũ Tuấn Anh, Nhà xuất bản: Lao Động Xã Hội, Xuất bản năm: 2020
6. Hướng dẫn làm việc với Android Studio, Tác giả: Trần Đức Cường, Nhà xuất bản: CNTT, Xuất bản năm: 2019