|  |
| --- |
|  |
| **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC** |
| **NGÀNH CÔNG NGHỆ THÔNG TIN** |
| **XÂY DỰNG ỨNG DỤNG QUẢN LÝ NHÀ HÀNG TRÊN NỀN TẢNG**  **ANDROID BẰNG SPRING BOOT VÀ MYSQL** |
|  |
|  |
| CBHD **: Ths. Vũ Duy Giang** |
| Sinh viên **: Nguyễn Xuân Thành** |
| Mã số sinh viên **: 2021603021** |
|  |
|  |
|  |
| Hà Nội – Năm … |
|  |

**MỞ ĐẦU**

Trong bối cảnh hội nhập kinh tế và sự phát triển vượt bậc của công nghệ, ngành dịch vụ ăn uống tại Việt Nam đang chứng kiến những thay đổi mạnh mẽ. Từ các nhà hàng sang trọng đến quán ăn bình dân, nhu cầu tối ưu hóa quy trình vận hành, nâng cao chất lượng phục vụ và quản lý hiệu quả nguồn lực ngày càng trở nên cấp thiết. Phương thức quản lý truyền thống với sổ sách, ghi chép thủ công đã bộc lộ nhiều hạn chế như tốn thời gian, dễ xảy ra sai sót, khó khăn trong việc tổng hợp dữ liệu và không đáp ứng kịp thời các yêu cầu ngày càng cao của khách hàng.

Trước thực trạng đó, việc ứng dụng công nghệ thông tin vào quản lý nhà hàng không còn là một lựa chọn mà đã trở thành một yếu tố then chốt quyết định sự cạnh tranh và phát triển bền vững. Các giải pháp phần mềm giúp tự động hóa quy trình order, thanh toán, quản lý thực đơn, nhân sự và tài chính đã chứng minh được hiệu quả vượt trội.

Nhận thấy được tầm quan trọng của việc ứng dụng công nghệ thông tin vào việc vận hành nhà hàng nên em đã nghiên cứu và xây dựng "Ứng dụng Quản lý Nhà hàng" với mục đích cung cấp một giải pháp công nghệ toàn diện, đa năng để giải quyết các thách thức mà các nhà hàng đang đối mặt.

Với sự kết hợp giữa các công nghệ hiện đại như Spring Boot cho backend và Firebase(Realtime Database) cho khả năng đồng bộ dữ liệu theo thời gian thực, mục tiêu mà em hướng đến cho đồ án là mang lại một hệ thống quản lý nhà hàng không chỉ hiệu quả, tin cậy mà còn linh hoạt và dễ dàng mở rộng trong tương lai. Đây sẽ là một công cụ hữu ích, đóng góp vào sự phát triển và chuyên nghiệp hóa của các mô hình kinh doanh ẩm thực nhỏ và vừa.

**2. Mục tiêu đề tài**

Ứng dụng hướng đến việc số hóa các nghiệp vụ cốt lõi, từ khâu phục vụ khách hàng trực tiếp đến công tác quản lý nội bộ. Cụ thể, hệ thống được thiết kế dành cho:

* **Nhân viên:** Cho phép nhân viên thực hiện các thao tác nhanh chóng và chính xác như **đăng nhập, mở bàn để order, order món ăn, tạo hóa đơn và thanh toán hóa đơn thông qua mã QR**, cùng với khả năng **xem lịch sử phục vụ qua hóa đơn** đã tạo. Điều này giúp giảm thiểu thời gian chờ đợi của khách, tăng cường hiệu quả làm việc và giảm thiểu sai sót.
* **Quản lý Nhà hàng:** Cung cấp đầy đủ các công cụ để người quản lý nắm bắt toàn bộ hoạt động kinh doanh, bao gồm **đăng nhập, đăng ký tài khoản quản lý, quản lý nhân viên, quản lý danh mục món ăn, quản lý món ăn, quản lý bàn, thống kê doanh thu và các số liệu quan trọng, quản lý hóa đơn chi tiết**. Các chức năng này hỗ trợ người quản lý đưa ra quyết định kịp thời, tối ưu hóa lợi nhuận và nâng cao chất lượng dịch vụ.

**3.Nội dung nghiên cứu**

- Nội dung nghiên cứu của đề tài này bao gồm :

* **Quy trình phát triển phần mềm:** Tuân thủ các giai đoạn chuẩn từ xác định yêu cầu, phân tích nghiệp vụ, thiết kế hệ thống, xây dựng ứng dụng đến kiểm thử để đảm bảo chất lượng sản phẩm.
* **Các công nghệ được sử dụng để xây dựng nên hệ thống**

**Android Java:** Nền tảng chính xây dựng ứng dụng di động phía nhân viên, cung cấp giao diện và hiệu năng.

**ViewModel và LiveData (Android Jetpack):** Các thành phần cốt lõi để quản lý dữ liệu UI theo vòng đời, giúp cập nhật giao diện an toàn và mượt mà.

**Firebase:** **Authentication:** Quản lý đăng nhập, đăng ký và xác thực người dùng.**Realtime Database:** Đồng bộ dữ liệu theo thời gian thực cho trạng thái bàn, order món, và thông báo, đảm bảo tương tác nhanh chóng.

**Spring Boot:** Framework chính xây dựng API backend, quản lý logic nghiệp vụ phức tạp và kết nối với cơ sở dữ liệu.

**MySQL:** Hệ quản trị cơ sở dữ liệu quan hệ được sử dụng để lưu trữ dữ liệu chính của hệ thống như thông tin nhà hàng, nhân viên, món ăn, danh mục, hóa đơn.

**4.Đối tượng và phạm vi nghiên cứu**

**Đối tượng nghiên cứu:**

* **Hệ thống Quản lý Nhà hàng:** Nghiên cứu tổng thể các nghiệp vụ quản lý bàn, order món, hóa đơn, thực đơn, nhân viên, thống kê.
* **Công nghệ phát triển:**
  + **Android Studio (Java):** Phát triển ứng dụng di động phía nhân viên (UI, logic).
  + **Spring Boot (Java):** Xây dựng hệ thống backend (API, logic nghiệp vụ).
  + **MySQL:** Cơ sở dữ liệu chính lưu trữ thông tin nhà hàng, nhân viên, món ăn, hóa đơn.
  + **Firebase (Auth, Realtime DB):** Quản lý xác thực người dùng và đồng bộ dữ liệu thời gian thực (trạng thái bàn, order, thông báo).
* **Mô hình MVVM (Android):** Nghiên cứu và áp dụng kiến trúc MVVM, ViewModel, LiveData để tối ưu hóa việc phát triển và bảo trì ứng dụng di động.

**Phạm vi nghiên cứu:**

* **Ứng dụng di động (phía nhân viên):**
  + Xây dựng trên nền tảng Android.
  + Các tính năng chính: đăng nhập, quản lý/mở bàn, order món, tạo/thanh toán hóa đơn (qua QR hoặc tiền mặt), xem lịch sử phục vụ.
  + Sử dụng Firebase Realtime DB cho trạng thái bàn và order tức thời.
* **Hệ thống Backend (API Server):**
  + Phát triển bằng Spring Boot.
  + Sử dụng MySQL làm cơ sở dữ liệu chính.
  + Quản lý thông tin nhà hàng, nhân viên, thực đơn, hóa đơn, voucher.
* **Giao diện Quản lý:** Xây dựng giao diện trên nền tảng Android cho phép quản lý: đăng nhập/đăng ký, quản lý nhân viên, thực đơn, hóa đơn, voucher, thống kê.
* **Công cụ phát triển:** Android Studio, IntelliJ IDEA, MySQL Workbench.
* **Ứng dụng thực tế:** Đáp ứng nhu cầu quản lý vận hành của các nhà hàng vừa và nhỏ.

**5.** **Kết quả mong muốn**

Đồ án hướng tới việc xây dựng một hệ thống quản lý nhà hàng hoàn chỉnh với các kết quả cụ thể:

1. Ứng dụng di động nhân viên (Android): Chức năng đăng nhập, quản lý/mở bàn, order, tạo/thanh toán hóa đơn (QR), xem lịch sử phục vụ.
2. Hệ thống Backend (Spring Boot & MySQL): API mạnh mẽ, quản lý dữ liệu chính (nhà hàng, nhân viên, món ăn, hóa đơn).
3. Giao diện quản lý: Cho phép quản lý thực hiện các tác vụ quản lý nhân viên, thực đơn, hóa đơn, và thống kê.
4. Tối ưu hóa quy trình nghiệp vụ: Giúp số hóa, tự động hóa và nâng cao hiệu quả vận hành nhà hàng.
5. Ứng dụng công nghệ: Chứng minh khả năng vận dụng hiệu quả Android Java, MVVM, ViewModel, LiveData, Spring Boot, MySQL và Firebase trong dự án thực tế.

Kết quả cuối cùng là tạo ra một hệ thống hiện đại, hiệu quả và đáng tin cậy cho việc quản lý và vận hành nhà hàng.

**CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI**

**1.1 Giới thiệu đề tài**

**1.1.1. Khảo sát thực tế**

Trong những năm gần đây, ngành công nghiệp dịch vụ ăn uống tại Việt Nam đã có sự tăng trưởng mạnh mẽ, kéo theo sự ra đời của hàng loạt nhà hàng, quán ăn với đa dạng phong cách và quy mô. Tuy nhiên, cùng với sự phát triển đó là những thách thức không nhỏ trong công tác quản lý và vận hành. Qua quá trình khảo sát thực tế tại một số nhà hàng, em nhận thấy các vấn đề phổ biến sau:

* Quản lý thủ công: Nhiều nhà hàng, đặc biệt là các cơ sở vừa và nhỏ, vẫn phụ thuộc nhiều vào việc ghi chép sổ sách, order bằng tay và tính toán hóa đơn thủ công. Điều này dẫn đến tình trạng chậm trễ trong khâu phục vụ, dễ xảy ra sai sót trong order và thanh toán, gây lãng phí thời gian của nhân viên và giảm trải nghiệm của khách hàng.
* Thiếu đồng bộ thông tin: Thông tin về tình trạng bàn, order của khách, hoặc thông báo từ bếp thường được truyền đạt qua giao tiếp trực tiếp hoặc các phương tiện thô sơ, dẫn đến thiếu đồng bộ, nhầm lẫn và khó khăn trong việc theo dõi tổng thể hoạt động.
* Hạn chế trong quản lý dữ liệu: Việc thiếu hệ thống lưu trữ và phân tích dữ liệu tập trung khiến quản lý khó khăn trong việc nắm bắt doanh thu chính xác, xác định món ăn bán chạy, quản lý kho nguyên vật liệu hoặc đưa ra các quyết định kinh doanh kịp thời dựa trên số liệu thực tế.

Những vấn đề trên đã ảnh hưởng trực tiếp đến hiệu quả hoạt động, chi phí vận hành và khả năng cạnh tranh của nhà hàng. Việc áp dụng công nghệ thông tin là giải pháp tối ưu để khắc phục những hạn chế này, mang lại lợi thế cạnh tranh trong thị trường đầy biến động.

**1.1.2. Giới thiệu đề tài**

Xuất phát từ những khảo sát thực tế và nhận thấy tầm quan trọng của việc ứng dụng công nghệ trong ngành dịch vụ ăn uống, đồ án "Ứng dụng Quản lý Nhà hàng" được ra đời nhằm cung cấp một giải pháp phần mềm toàn diện, hiện đại và hiệu quả. Đề tài tập trung vào việc số hóa và tự động hóa các quy trình nghiệp vụ cốt lõi của một nhà hàng, từ khâu phục vụ khách hàng đến công tác quản lý nội bộ.

Hệ thống được thiết kế bao gồm hai phân hệ chính:

* Phân hệ dành cho Nhân viên (ứng dụng di động Android): Cung cấp các công cụ trực quan để nhân viên thực hiện các tác vụ hàng ngày như:

- Đăng nhập,

- Quản lý và mở bàn,

- Order món ăn trực tiếp từ thiết bị di động,

- Tạo hóa đơn và thanh toán tiện lợi (bao gồm tích hợp thanh toán qua mã QR),

- Xem lại lịch sử phục vụ thông qua các hóa đơn đã tạo.

Điều này giúp tối ưu hóa tốc độ phục vụ, giảm thiểu sai sót và nâng cao trải nghiệm của khách hàng.

* Phân hệ dành cho Quản lý Nhà hàng:

- Đăng nhập/đăng ký tài khoản,

- Quản lý nhân viên,

- Quản lý danh mục

- Quản lý món ăn

- Xem báo cáo thống kê (doanh thu, món ăn bán chạy),

- Quản lý hoá đơn.

Các chức năng này hỗ trợ đắc lực cho việc ra quyết định kinh doanh, kiểm soát chi phí và tối ưu hóa lợi nhuận.

Đồ án được xây dựng trên nền tảng công nghệ mạnh mẽ và hiện đại: Spring Boot cho hệ thống backend vững chắc, MySQL cho cơ sở dữ liệu quan hệ tin cậy, và đặc biệt là Firebase (Authentication và Realtime Database) để đảm bảo tính năng xác thực người dùng và đồng bộ dữ liệu theo thời gian thực. Đối với ứng dụng di động phía nhân viên, chúng tôi sử dụng Android Java kết hợp với kiến trúc MVVM (Model-View-ViewModel) cùng các thành phần ViewModel và LiveData nhằm tối ưu hóa hiệu suất và khả năng bảo trì.

Với mục tiêu đem lại một công cụ quản lý hiệu quả, "Ứng dụng Quản lý Nhà hàng" không chỉ giải quyết các vấn đề hiện tại mà còn góp phần nâng cao năng lực cạnh tranh, thúc đẩy sự phát triển chuyên nghiệp cho ngành dịch vụ ăn uống.

**1.2 Công nghệ sử dụng**

**1.2.1 Ngôn ngữ Java**

Java là một ngôn ngữ lập trình hướng đối tượng, đa nền tảng, được phát triển bởi Sun Microsystems (nay thuộc Oracle). Từ những ngày đầu ra mắt hệ điều hành Android, Java đã được Google chọn làm ngôn ngữ chính thức và ưu tiên hàng đầu để phát triển ứng dụng di động.

Sự phổ biến của Java trong lập trình Android đến từ nhiều yếu tố:

* Tính ổn định và trưởng thành: Java là một ngôn ngữ đã có tuổi đời lâu và được kiểm chứng, sở hữu một hệ sinh thái mạnh mẽ với hàng ngàn thư viện và framework hỗ trợ.
* Hỗ trợ rộng rãi: Google cung cấp bộ công cụ phát triển phần mềm (SDK) Android được viết chủ yếu bằng Java, cùng với tài liệu phong phú và cộng đồng lập trình viên lớn. Điều này giúp các nhà phát triển dễ dàng tìm kiếm tài nguyên và hỗ trợ khi gặp vấn đề.
* Đa nền tảng: Mặc dù được sử dụng chủ yếu cho Android, nhưng bản chất "Write once, run anywhere" (Viết một lần, chạy mọi nơi) của Java cho phép các kiến thức và kỹ năng có thể áp dụng cho nhiều môi trường khác.
* Hiệu suất: Với sự tối ưu hóa của máy ảo Dalvik (hoặc ART hiện nay) trên Android, các ứng dụng Java có thể chạy với hiệu suất tốt trên nhiều thiết bị.

Mặc dù Kotlin đang dần trở thành ngôn ngữ ưu tiên mới cho Android, Java vẫn đóng vai trò quan trọng và được sử dụng rộng rãi trong hàng triệu ứng dụng hiện có. Java cũng được hỗ trợ bởi hệ sinh thái phong phú bao gồm các thư viện, framework và cộng đồng người dùng lớn mạnh, giúp quá trình học tập và phát triển ứng dụng diễn ra dễ dàng hơn.

**Ưu điểm của ngôn ngữ Java so với Kotlin:**

* **Hệ sinh thái và cộng đồng lớn mạnh:** Java đã tồn tại từ lâu với một cộng đồng lập trình viên khổng lồ, tài liệu phong phú, và vô số thư viện, framework đã được kiểm chứng. Điều này giúp dễ dàng tìm kiếm giải pháp cho các vấn đề.
* **Tính ổn định và trưởng thành:** Là ngôn ngữ đã được sử dụng rộng rãi trong nhiều năm, Java rất ổn định và đáng tin cậy cho các dự án lớn, phức tạp.
* **Khả năng tương thích ngược:** Java có khả năng tương thích ngược tốt, cho phép duy trì và mở rộng các dự án Android cũ một cách dễ dàng.
* **Hiệu suất:** Với các tối ưu hóa của JVM (Java Virtual Machine) và ART (Android Runtime), Java cung cấp hiệu suất tốt cho các ứng dụng Android.

**Nhược điểm so với Kotlin:**

* **Cú pháp dài dòng (Boilerplate code):** Java yêu cầu nhiều mã lặp lại (boilerplate code) cho các tác vụ đơn giản như tạo getter/setter, equals(), hashCode(), toString(), làm cho mã nguồn trở nên dài và khó đọc hơn.
* **Dễ gặp lỗi NullPointerException:** Java không có tính năng an toàn null tích hợp, dẫn đến nguy cơ gặp lỗi NullPointerException (NPE) rất phổ biến trong quá trình chạy ứng dụng.
* **Phức tạp hơn cho các tính năng hiện đại:** Việc triển khai một số tính năng hiện đại như lập trình hàm hoặc coroutines trong Java có thể phức tạp hơn so với Kotlin.
* **Thời gian biên dịch (full build) đôi khi chậm:** Mặc dù biên dịch tăng dần (incremental build) nhanh, nhưng thời gian biên dịch toàn bộ dự án Java đôi khi có thể chậm hơn.

**1.2.2 Giới thiệu về công cụ Android Studio**

**-** Android Studio là một môi trường tích hợp phát triển (Integrated Development Environment – IDE) được phát triển bởi Google dành cho việc phát triển ứng dụng trên nền tảng Android. Nó cung cấp một loạt các công cụ và tính năng để giúp nhà phát triển xây dựng và triển khai ứng dụng Android một cách dễ dàng.Công cụ này cung cấp một giao diện trực quan và các trình biên dịch, trình gỡ lỗi, trình tạo giao diện người dùng, trình quản lý phiên bản và nhiều công cụ khác. Nó hỗ trợ viết code trong ngôn ngữ Java hoặc Kotlin và tích hợp sẵn các thư viện và công cụ phát triển Android.

Một trong những tính năng quan trọng của  công cụ này là khả năng sử dụng Gradle, một hệ thống quản lý dự án mạnh mẽ. Gradle cho phép bạn quản lý phụ thuộc, xây dựng, kiểm thử và đóng gói ứng dụng Android một cách linh hoạt và hiệu quả.

Ưu điểm

* Đây là môi trường phát triển phần mềm chính thức của Google, đây cũng chính là chủ sở hữu của hệ điều hành Android.
* Công cụ này cung cấp một loạt các công cụ phát triển và tính năng hỗ trợ cho quá trình phát triển ứng dụng Android. Điều này bao gồm trình biên dịch, trình gỡ lỗi, trình quản lý dự án, thiết kế giao diện và nhiều tính năng khác để tăng năng suất và hiệu quả của nhà phát triển.
* Giao diện người dùng thân thiện và dễ sử dụng, giúp tăng hiệu suất và giảm thời gian cần thiết để phát triển ứng dụng.
* Hỗ trợ bởi một số tài liệu tham khảo và hướng dẫn phong phú. Có sẵn các tài liệu chính thức từ Google, cùng với các diễn đàn lập trình viên Android và các nguồn tài liệu trực tuyến khác để giúp nhà phát triển tìm hiểu và giải quyết các vấn đề phát triển.
* Công cụ này được hỗ trợ thông qua các khóa học đào tạo về lập trình Android cơ bản và nâng cao. Các khóa học này giúp nhà phát triển nắm vững các khái niệm và kỹ năng cần thiết để phát triển ứng dụng Android chất lượng.

Nhược điểm

* Đây là công cụ chiếm lượng lớn dữ liệu trong không gian bộ nhớ máy tính nếu được cài đặt
* Việc kiểm tra hoạt động thông qua giả lập gây đơ,lag, giật máy và tiêu tốn pin

**1.2.3 Firebase**

**-** Firebase là một nền tảng phát triển ứng dụng di động và web được Google cung cấp. Nó cung cấp một tập hợp các công cụ và dịch vụ giúp lập trình viên xây dựng, vận hành và mở rộng ứng dụng một cách nhanh chóng mà không cần quản lý cơ sở hạ tầng phức tạp. Firebase đặc biệt mạnh mẽ khi sử dụng với các ứng dụng di động như Android và iOS, nhưng cũng hỗ trợ tốt cho phát triển web.

- Các dịch vụ nổi bật của Firebase

* Realtime Database
* Cloud Firestore
* Authentication
* Cloud Storage
* Cloud Messaging
* Remote Config
* In-App Messaging

**Ưu điểm:**

* **Tốc độ phát triển nhanh:** Firebase cung cấp các SDK dễ sử dụng cho nhiều nền tảng (Android, iOS, Web, Unity, C++), giúp các nhà phát triển nhanh chóng tích hợp các chức năng backend phức tạp mà không cần phải viết và quản lý máy chủ riêng.
* **Đồng bộ dữ liệu thời gian thực (Realtime Database):** Đây là một trong những điểm mạnh nổi bật nhất. Realtime Database cho phép đồng bộ hóa dữ liệu gần như tức thì giữa các client và backend, rất lý tưởng cho các ứng dụng yêu cầu cập nhật liên tục như ứng dụng chat, game nhiều người chơi, hoặc như trong trường hợp của bạn là quản lý trạng thái bàn và order.
* **Xác thực mạnh mẽ (Authentication):** Firebase Authentication cung cấp giải pháp xác thực toàn diện, hỗ trợ nhiều phương thức đăng nhập phổ biến (email/mật khẩu, Google, Facebook, Twitter, số điện thoại, v.v.), giúp giảm đáng kể công sức phát triển chức năng đăng nhập.
* **Khả năng mở rộng tự động (Scalability):** Firebase được thiết kế để tự động mở rộng theo nhu cầu của ứng dụng, loại bỏ gánh nặng quản lý cơ sở hạ tầng cho nhà phát triển.
* **Chi phí khởi đầu thấp (Free Tier):** Firebase cung cấp một gói miễn phí hào phóng (Spark Plan) cho phép phát triển và thử nghiệm ứng dụng mà không tốn chi phí ban đầu.
* **Tích hợp với hệ sinh thái Google:** Dễ dàng tích hợp với các dịch vụ khác của Google Cloud và các công cụ phát triển của Google.
* **Không yêu cầu quản lý máy chủ:** Là dịch vụ BaaS, Firebase giúp bạn không cần phải lo lắng về việc cấu hình, duy trì hay nâng cấp máy chủ backend.

**Nhược điểm:**

* **Phụ thuộc vào Google:** Việc sử dụng Firebase đồng nghĩa với việc ứng dụng của bạn sẽ phụ thuộc chặt chẽ vào hệ sinh thái và chính sách của Google. Việc chuyển đổi sang một nền tảng khác trong tương lai có thể phức tạp.
* **Kiểm soát hạn chế đối với Backend:** Vì là một dịch vụ BaaS, bạn sẽ có ít quyền kiểm soát hơn đối với logic backend, máy chủ, và cơ sở hạ tầng so với việc tự xây dựng backend tùy chỉnh. Điều này có thể là một hạn chế đối với các yêu cầu nghiệp vụ rất phức tạp hoặc tùy chỉnh cao.
* **Chi phí có thể tăng cao khi mở rộng:** Mặc dù có gói miễn phí, nhưng khi ứng dụng của bạn phát triển và có lượng người dùng lớn, chi phí sử dụng Firebase (đặc biệt là cho Realtime Database và Cloud Firestore) có thể tăng lên đáng kể và đôi khi khó dự đoán.
* **Chỉ hỗ trợ NoSQL:** Firebase Realtime Database là cơ sở dữ liệu NoSQL, không có khả năng truy vấn phức tạp hoặc hỗ trợ các quan hệ dữ liệu mạnh mẽ như cơ sở dữ liệu quan hệ (SQL). Điều này có thể gây khó khăn nếu dữ liệu của bạn có cấu trúc phức tạp hoặc cần các truy vấn JOIN.
* **Khó khăn trong việc di chuyển dữ liệu (Vendor Lock-in):** Việc xuất dữ liệu từ Firebase để di chuyển sang một nền tảng khác có thể không đơn giản, tiềm ẩn rủi ro bị "khóa chặt" vào Firebase.
* **Hạn chế truy vấn và lọc dữ liệu (Realtime Database):** Realtime Database có các hạn chế về khả năng truy vấn và lọc dữ liệu. Đối với các truy vấn phức tạp hoặc yêu cầu tìm kiếm toàn văn, bạn có thể cần kết hợp với các dịch vụ khác hoặc tái cấu trúc dữ liệu.
* **Chỉ lưu trữ dạng JSON:** Dữ liệu trong Realtime Database được lưu trữ dưới dạng JSON cây (JSON tree). Điều này yêu cầu cách tiếp cận khác trong việc thiết kế dữ liệu so với cơ sở dữ liệu quan hệ truyền thống.

**1.2.4 Spring Boot**

- Spring Boot là một framework mạnh mẽ, đơn giản hóa quá trình phát triển các ứng dụng Java bằng cách tự động hóa nhiều cấu hình trong Spring Framework. Đặc biệt với việc xây dựng các dịch vụ web RESTful, Spring Boot cung cấp các công cụ và tính năng hỗ trợ lập trình viên dễ dàng tạo các API nhanh chóng và linh hoạt mà không cần phải cấu hình phức tạp.

Trong Spring Boot, các API RESTful thường được phát triển theo mô hình kiến trúc REST (Representational State Transfer), giúp giao tiếp dễ dàng với các hệ thống khác nhau thông qua giao thức HTTP, sử dụng các phương thức như GET, POST, PUT, và DELETE.

**Ưu Điểm :**

Cấu hình đơn giản: Với Spring Boot, các thiết lập cấu hình phức tạp được tự động hóa nhờ tính năng Spring Boot Auto-Configuration và các starter như spring-boot-starter-web giúp tiết kiệm thời gian khi thiết lập dự án RESTful API.

Hỗ trợ mạnh mẽ cho RESTful: Spring Boot cung cấp các annotation và công cụ tối ưu cho RESTful API như @RestController, @RequestMapping, @GetMapping, @PostMapping,… giúp dễ dàng phát triển các dịch vụ RESTful.

Tích hợp dễ dàng với các công nghệ khác: Spring Boot dễ dàng tích hợp với các thành phần khác của Spring như Spring Security để bảo mật API, Spring Data JPA để quản lý dữ liệu, và Spring Cloud để xây dựng các hệ thống phân tán.

Đơn giản hóa quản lý lỗi: Cung cấp khả năng xử lý và quản lý lỗi thông qua @ExceptionHandler và @ControllerAdvice, giúp định nghĩa rõ ràng các phản hồi lỗi đến client.

**Nhược Điểm :**

Hiệu suất: So với một số framework nhẹ hơn, Spring Boot có thể chậm hơn vì tính năng và tính linh hoạt cao.

Phụ thuộc vào hệ sinh thái Spring: Sử dụng Spring Boot đồng nghĩa với việc gắn bó sâu sắc với hệ sinh thái Spring, điều này có thể gây khó khăn khi cần chuyển đổi sang các công nghệ khác

**1.2.5 My SQL**

- MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở phổ biến được phát triển và duy trì bởi Oracle Corporation. Nó được thiết kế để quản lý các cơ sở dữ liệu quan hệ, giúp lưu trữ và truy vấn dữ liệu hiệu quả. MySQL được sử dụng rộng rãi trong các ứng dụng web, dịch vụ trực tuyến và nhiều ứng dụng doanh nghiệp nhờ vào khả năng mạnh mẽ và tính linh hoạt của nó.

- MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS), trong đó dữ liệu được tổ chức thành các bảng (tables). Mỗi bảng gồm các hàng (rows) và các cột (columns), và mỗi dữ liệu được thêm vào bảng phải tuân theo cấu trúc đã quy định của bảng đó. MySQL sử dụng ngôn ngữ truy vấn có cấu trúc (SQL) để thao tác với dữ liệu, bao gồm việc thêm, sửa, xóa và truy vấn dữ liệu.

**Ưu điểm**

* Tránh rùng lặp dữ liệu, dữ liệu không đồng bộ, không an toàn, lọc và tìm kiếm thông tin không dễ dàng
* Dễ dàng truy cập database và lấy thông tin khi cần.
* Thông tin được cập nhật trên cơ sở dữ liệu thì người dùng lấy dữ liệu, dữ liệu sẽ được đảm bảo đồng bộ hoàn toàn.
* Thông tin được quản lý ở một nơi duy nhất, đảm bảo việc bảo mật dữ liệu, tránh rò rỉ thông tin. Ngoài ra có thể phân quyền cho người dùng, ai có thể truy cập dữ liệu
* Việc tìm kiếm thông tin cũng vô cùng dễ dàng khi dữ liệu đã được quản lsy bởi DBMS, chúng có thể tìm kiếm, gom nhóm các thông tin lại với nhau bằng cách dùng câu truy vấn.

**Nhược điểm**

* Việc sử dụng cơ sở dữ liệu cũng có một số nhược điểm sau:
* Việc sử dụng chung cơ sở dữ liệu khi xảy ra sự cố, toàn bộ dữ liệu sẽ bị mất hoàn toàn.
* Việc dùng chung cơ sở dữ liệu khi có quá nhiều truy vấn sẽ gây nghẽn, sập hệ thống.

**1.3 Tổng kết chương 1**

Trong chương 1, em đã tiến hành khảo sát thực trạng hoạt động của các nhà hàng hiện nay, đặc biệt tập trung vào những thách thức trong quản lý thủ công, thiếu đồng bộ thông tin, và hạn chế trong việc quản lý dữ liệu. Qua đó, chúng em đã nhận diện rõ ràng các vấn đề mà một hệ thống quản lý nhà hàng cần phải giải quyết để tối ưu hóa quy trình vận hành và nâng cao hiệu quả kinh doanh.

Dựa trên kết quả khảo sát, đồ án "Ứng dụng Quản lý Nhà hàng" được đề xuất với mục tiêu xây dựng một giải pháp phần mềm toàn diện. Đề tài tập trung vào việc số hóa các nghiệp vụ cốt lõi, chia thành hai phân hệ chính: một ứng dụng di động Android cho nhân viên với các chức năng như mở bàn, order món, tạo và thanh toán hóa đơn (hỗ trợ QR), xem lịch sử phục vụ và một giao diện quản lý cho phép người quản lý thực hiện các tác vụ quản lý nhân viên, thực đơn, hóa đơn, voucher và xem thống kê.

Chương này cũng đã trình bày tổng quan về các phương pháp nghiên cứu được áp dụng, bao gồm việc tuân thủ quy trình phát triển phần mềm chuẩn và nghiên cứu sâu các công nghệ chủ chốt. Các công nghệ này bao gồm Android Java cho ứng dụng di động; Spring Boot và MySQL cho hệ thống backend vững chắc; và Firebase (Authentication, Realtime Database) để xử lý xác thực người dùng và đồng bộ dữ liệu thời gian thực.

Tóm lại, Chương 1 đã xác định rõ mục tiêu, đối tượng và phạm vi nghiên cứu của đề tài, đồng thời giới thiệu các công cụ, phương pháp và nền tảng kỹ thuật sẽ được sử dụng trong suốt quá trình phát triển ứng dụng. Đây là cơ sở vững chắc để chúng em triển khai chi tiết các chức năng và thành phần của hệ thống trong các chương tiếp theo, với mong muốn tạo ra một sản phẩm hữu ích, giải quyết được nhu cầu thực tế của ngành dịch vụ nhà hàng.

**Chương 2: Phân tích và thiết kế hệ thống**

**2.1 Xác định các tác nhân,ca sử dụng**

**2.1.1 Tác nhân**

|  |  |
| --- | --- |
| **Tác nhân** | **Ca sử dụng** |
| **Nhân viên** | **Đăng nhập**  **Quản lý trạng thái bàn**  **Order món ăn cho khách**  **Tạo hoá đơn và thanh toán**  **Xem lịch sử phục vụ** |
| **Quản lý** | **Đăng nhập**  **Đăng ký nhà hàng**  **Quản lý danh mục món ăn**  **Quản lý nhân viên**  **Quản lý món ăn**  **Thống kê**  **Quản lý bàn**  **Quản lý hoá đơn** |

**2.1.2 Use Case**

**2.2 Biểu đồ Use case**

**2.3 Mô tả chi tiết Use Case**

**2.3.1** **Mô tả use case Đăng nhập**

* **Tên Use Case**: Đăng nhập
* **Mô tả vắn tắt**: Use case này cho phép người dùng đăng nhập vào hệ thống.
* Luồng cơ bản:

1. Use case này bắt đầu khi người dùng chọn vai trò và nhập thông tin (tài khoản và mật khẩu) vào form đăng nhập và nhấn nút "Đăng nhập".Hệ thống xác thực thông tin đăng nhập với cơ sở dữ liệu.
2. Nếu thông tin hợp lệ, hệ thống chuyển hướng người dùng đến trang chủ.Use case kết thúc

* Luồng rẽ nhánh:

1. Nếu thông tin đăng nhập không hợp lệ, hệ thống hiển thị thông báo "Thông tin đăng nhập không chính xác" và yêu cầu người dùng nhập lại.
2. Nếu thông tin đăng nhập hợp lệ mà tài khoản mà không chọn đúng vai trò (nhân viên,quản lý) thì hiển thị thông báo ‘Thông tin đăng nhập không chính xác
3. Nếu không kết nối được với cơ sở dữ liệu, hệ thống hiển thị thông báo lỗi "Lỗi kết nối" và kết thúc use case.

* Các yêu cầu đặc biệt: Không có.
* Tiền điều kiện: Người dùng đã đăng ký tài khoản hoặc đã được tạo tài khoản( Đối với nhân viên)
* Hậu điều kiện: Không có.
* Điểm mở rộng: Không có.

**2.3.2 Mô tả Use Case Quản lý bàn (Nhân viên)**

* **Tên Use Case:** Quản lý bàn
* **Mô tả vắn tắt:** Use case này cho phép nhân viên xem trạng thái bàn, mở bàn khi trống, hoặc truy cập bàn đang phục vụ nếu là người chịu trách nhiệm, để thực hiện các nghiệp vụ tiếp theo như order món.
* **Luồng các sự kiện:** 
  + **Luồng cơ bản:** 
    1. Use case này bắt đầu khi nhân viên chọn chức năng "Bàn" ở menu. Hệ thống hiển thị danh sách các bàn và trạng thái hiện tại của chúng (ví dụ: "Trống", "Đang phục vụ").
    2. Nhân viên chạm (click/chọn) vào một bàn cụ thể. Hệ thống kiểm tra trạng thái của bàn đó.
    3. Nếu bàn đang ở trạng thái "Trống":

a. Hệ thống cho phép nhân viên thực hiện thao tác "Mở bàn".

b. Khi nhân viên xác nhận "Mở bàn", hệ thống sẽ cập nhật trạng thái của bàn đó thành "Đang phục vụ" và ghi lại ID (mã định danh) của nhân viên hiện tại vào trường firebaseUID của bàn. . Hệ thống chuyển hướng nhân viên đến giao diện "Order món ăn" để bắt đầu phục vụ bàn này. Use case kết thúc.

* + 1. Nếu bàn đang ở trạng thái "Đang phục vụ":

**a.** Hệ thống kiểm tra ID của nhân viên hiện tại với firebaseUID đã lưu cho bàn đó.

**b.** Nếu ID của nhân viên hiện tại TRÙNG với firebaseUID : Hệ thống cho phép nhân viên truy cập vào bàn và chuyển hướng đến giao diện "Order món ăn" hoặc "Tạo hóa đơn" để tiếp tục phục vụ bàn đó. Use case kết thúc.

**c.** Nếu ID của nhân viên hiện tại KHÔNG TRÙNG với ID\_NhanVienPhucVu: Hệ thống hiển thị thông báo "Không thể mở bàn." Use case kết thúc.

* + **Luồng rẽ nhánh:** 
    1. Tại bất kỳ bước nào, nếu xảy ra lỗi kết nối với cơ sở dữ liệu hoặc hệ thống không thể lấy/cập nhật thông tin bàn, hệ thống sẽ hiển thị thông báo lỗi "Lỗi hệ thống, vui lòng thử lại sau."
* **Các yêu cầu đặc biệt:** Không có.
* **Tiền điều kiện:** Nhân viên đã đăng nhập vào hệ thống và có kết nối mạng.
* **Hậu điều kiện:** 
  + Nếu thành công khi bàn "Trống": Trạng thái bàn được cập nhật thành "Đang phục vụ" và ID\_NhanVienPhucVu được lưu lại.
  + Nếu thành công khi bàn "Đang phục vụ" và trùng ID: Nhân viên được phép truy cập bàn.
* **Điểm mở rộng:** Không có.

**2.3.3 Mô tả Use Case Order món ăn**

* **Tên Use Case:** Order món ăn
* **Mô tả vắn tắt:** Use case này cho phép nhân viên chọn và order món ăn cho khách theo bàn, tạo hoặc cập nhật đơn hàng
* **Luồng các sự kiện:** 
  + **Luồng cơ bản:** 
    1. Use case này bắt đầu khi nhân viên đang ở màn hình order món ăn cho một bàn cụ thể (đã được truy cập hợp lệ từ Use Case Quản lý bàn). Hệ thống hiển thị danh sách các món ăn có sẵn theo danh mục.
    2. Nhân viên chọn các món ăn mà khách order và nhập số lượng tương ứng cho mỗi món. Các món đã chọn sẽ được hiển thị trong màn Order.
    3. Sau khi đã chọn xong tất cả các món cho lượt order hiện tại, nhân viên ấn vào nút "Xác nhận thêm món".

4.1 Hệ thống kiểm tra trạng thái đơn hàng của bàn trên Firebase:

* + - 1. Nếu bàn hiện tại CHƯA CÓ đơn hàng đang chờ trên Firebase (lần order đầu tiên hoặc đơn hàng cũ đã được lưu vào database):Hệ thống sẽ tạo một bản ghi đơn hàng mới trong bảng DonHang
      2. Sau khi tạo đơn hàng thành công, hệ thống sẽ thêm các món ăn đã chọn vào bảng MonAnTrongDonHang chứa maDonHang
      3. Hệ thống sẽ ghi maDonHang này lên Firebase cho bàn tương ứng, đánh dấu bàn đó có một đơn hàng đang được xử lý.
      4. Hệ thống xóa danh sách món ăn đã chọn tạm thời của bàn đó trên Firebase (nếu có để phục vụ việc hiển thị cho bếp).
      5. Hệ thống hiển thị thông báo "Order đã được thêm vào đơn hàng mới."

4.2 Nếu bàn hiện tại ĐÃ CÓ một maDonHang đang chờ trên Firebase:

a. Hệ thống sẽ lấy maDonHang đã có từ Firebase cho bàn đó.

b. Hệ thống sẽ thêm các món ăn đã chọn vào bảng MonAnTrongDonHang, liên kết với ID\_DonHang đang có.

c. Hệ thống xóa danh sách món ăn đã chọn tạm thời của bàn đó trên Firebase.

d. Hệ thống hiển thị thông báo "Order đã được cập nhật vào đơn hàng hiện có."

5. Use case kết thúc.

* 1. **Luồng rẽ nhánh:** 
     1. Số lượng không hợp lệ: Nếu nhân viên chọn số lượng bé hơn 0,hệ thống sẽ xoá món đó khỏi danh sách tạm thời trên Firebase.
     2. Lỗi kết nối/Hệ thống: Nếu không kết nối được với cơ sở dữ liệu MySQL hoặc Firebase, hoặc xảy ra lỗi trong quá trình ghi/đọc dữ liệu, hệ thống sẽ hiển thị thông báo lỗi "Lỗi hệ thống, vui lòng thử lại sau."

1. **Các yêu cầu đặc biệt:** Không có.
2. **Tiền điều kiện:** Nhân viên đã đăng nhập vào hệ thống và đã truy cập thành công vào màn hình order của một bàn cụ thể (bàn đang ở trạng thái "Đang phục vụ" và nhân viên có quyền truy cập).
3. **Hậu điều kiện:** 
   1. Nếu thành công: Một bản ghi đơn hàng mới được tạo trong MySQL và các món ăn được thêm vào chi tiết đơn hàng, HOẶC các món ăn được thêm vào đơn hàng hiện có trong MySQL. Danh sách món đã chọn tạm thời trên Firebase được xóa.
4. **Điểm mở rộng:** Không có.

**2.3.4 Mô tả Use Case Tạo Hóa đơn**

* **Tên Use Case:** Tạo và thanh toán hóa đơn
* **Mô tả vắn tắt:** Use case này cho phép nhân viên tạo hóa đơn và thực hiện thanh toán.
  + **Luồng cơ bản:**
    1. Use case này bắt đầu khi nhân viên chọn một bàn và chọn chức năng "Thanh toán". Hệ thống tạo hóa đơn từ order.
    2. Nhân viên chọn phương thức thanh toán và xác nhận thanh toán.

a) Nếu chọn phương thức tiền mặt thì tạo hoá đơn với thuộc tính trạng thái là “Đã thanh toán”

b) Nếu chọn phương thức chuyển khoản thì tạo hoá đơn với thuộc tính là “Chưa thanh toán”

* + 1. Hệ thống cập nhật trạng thái hóa đơn và bàn. Use case kết thúc.
  + **Luồng rẽ nhánh:**
    1. Nếu mã voucher không hợp lệ, hệ thống hiển thị thông báo lỗi.
    2. Nếu quá trình thanh toán gặp lỗi, hệ thống hiển thị thông báo lỗi.
* **Các yêu cầu đặc biệt:** Không có.
* **Tiền điều kiện:** Nhân viên đã đăng nhập và đã chọn một bàn có order và bàn đó do nhân viên đó phụ trách.
* **Hậu điều kiện:** Hóa đơn được tạo và thông tin thanh toán được lưu. Trạng thái bàn được cập nhật.
* **Điểm mở rộng:** Không có.

**2.3.5 Mô tả Use Case Đăng ký Nhà hàng**

* **Tên Use Case:** Đăng ký Nhà hàng
* **Mô tả vắn tắt:** Use case này cho phép một chủ nhà hàng hoặc người đại diện đăng ký thông tin nhà hàng của mình để sử dụng hệ thống quản lý ứng dụng. Khi đăng ký thành công, một tài khoản quản lý ban đầu cho nhà hàng đó cũng sẽ được tạo ra.
* **Luồng các sự kiện:** 
  + **Luồng cơ bản:** 
    1. Use case bắt đầu khi người dùng (chủ nhà hàng/đại diện) truy cập vào màn hình "Đăng ký Nhà hàng" trên ứng dụng
    2. Người dùng nhập các thông tin cần thiết về nhà hàng, bao gồm:
       - Thông tin nhà hàng: Tên nhà hàng, Địa chỉ, Số điện thoại liên hệ, Email nhà hàng.
       - Thông tin tài khoản quản lý ban đầu: Email đăng nhập (cho tài khoản quản lý đầu tiên), Mật khẩu, Xác nhận mật khẩu, Tên đầy đủ của người quản lý.
    3. Người dùng nhấn nút "Đăng ký Nhà hàng".
    4. Hệ thống thực hiện kiểm tra dữ liệu đầu vào (ví dụ: định dạng email, độ dài mật khẩu, mật khẩu và xác nhận mật khẩu phải trùng khớp, tên nhà hàng không được trống).
    5. Nếu dữ liệu hợp lệ, hệ thống gửi yêu cầu đến Backend Spring Boot.
    6. Backend Spring Boot thực hiện các bước sau:

**a**. Kiểm tra xem tên nhà hàng hoặc email quản lý đã tồn tại trong hệ thống chưa.

**b.** Nếu chưa, tạo một bản ghi nhà hàng mới trong bảng NhaHang của MySQL, gán một MaNH (Mã Nhà hàng) duy nhất.

**c.** Sử dụng Email đăng nhập và Mật khẩu cung cấp để tạo một tài khoản người dùng trên Firebase Authentication.

**d.**Hệ thống hiển thị thông báo "Đăng ký thành công." và chuyển hướng người dùng đến màn hình đăng nhập. Use case kết thúc.

* + **Luồng rẽ nhánh:** 
    1. Thông tin nhập vào không hợp lệ: Tại bước 4, nếu thông tin nhập vào không hợp lệ (ví dụ: tên nhà hàng trống, email sai định dạng, mật khẩu quá ngắn, mật khẩu xác nhận không khớp), hệ thống hiển thị thông báo lỗi tương ứng và yêu cầu người dùng nhập lại.
    2. Tên nhà hàng/Email quản lý đã tồn tại: Tại bước 6a, nếu tên nhà hàng hoặc email quản lý đã tồn tại trong hệ thống, hệ thống hiển thị thông báo lỗi "Tên nhà hàng hoặc Email quản lý đã được đăng ký. Vui lòng kiểm tra lại thông tin." Người dùng được yêu cầu sửa đổi thông tin.
    3. Lỗi hệ thống/Kết nối: Tại bất kỳ bước nào từ 5 đến 7, nếu có lỗi kết nối mạng hoặc lỗi hệ thống xảy ra (lỗi khi tạo bản ghi trong MySQL, lỗi khi tạo tài khoản Firebase), hệ thống hiển thị thông báo lỗi chung "Đã xảy ra lỗi"
* **Các yêu cầu đặc biệt:** Không có.
* **Tiền điều kiện:** Người dùng có kết nối mạng**.**
* **Hậu điều kiện:** Nếu đăng ký thành công, một nhà hàng mới và tài khoản quản lý ban đầu sẽ được tạo và lưu trữ trong hệ thống (MySQL và Firebase Authentication). Người dùng có thể sử dụng tài khoản đó để đăng nhập và bắt đầu quản lý nhà hàng của mình.
* **Điểm mở rộng:** Không có.

**2.3.6 Mô tả Use Case Quản lý danh mục món ăn**

* **Tên Use Case:** Quản lý danh mục món ăn
* **Mô tả vắn tắt:** Người quản lý có thể thêm, sửa, xóa các danh mục món ăn.
* **Luồng các sự kiện** 
  + **Luồng cơ bản:** 
    1. Use case bắt đầu khi người quản lý chọn chức năng "Quản lý danh mục món ăn" trên menu quản trị. Hệ thống lấy tất cả thông tin về danh mục món ăn từ bảng DanhMuc trong cơ sở dữ liệu và hiển thị danh sách danh mục lên màn hình.
    2. **Thêm thông tin danh mục:**

**a.** Người quản lý kích vào nút "Thêm" trên cửa sổ quản lý danh mục. Hệ thống hiển thị màn hình yêu cầu nhập thông tin về danh mục.

b. Người quản lý nhập tên danh mục và kích vào nút "Thêm". Hệ thống sẽ tạo một MaDanhMuc mới, cập nhật bảng DanhMuc và hiển thị danh sách danh mục đã được cập nhật.

* + 1. **Sửa thông tin danh mục:**

**a.** Người quản lý kích vào nút "Sửa" trên một hàng danh mục. Hệ thống lấy tên cũ của danh mục được chọn.

b. Người quản lý nhập tên mới cho danh mục đó sau đó kích vào nút "Cập nhật". Hệ thống sẽ cập nhật tên của danh mục được chọn trong bảng DanhMuc và hiển thị danh sách danh mục đã được cập nhật.

* + 1. **Xóa thông tin danh mục:**

**a.** Người quản lý kích vào nút "Xóa" trên một dòng danh mục. Hệ thống sẽ hiển thị một màn hình yêu cầu xác nhận xóa.

**b.** Người quản lý kích vào nút "Đồng ý". Hệ thống sẽ xóa danh mục được chọn khỏi bảng DanhMuc và hiển thị danh sách danh mục đã được cập nhật.

* + **Luồng rẽ nhánh:** 
    1. Tại bước 4b trong luồng cơ bản, nếu người quản lý kích vào nút "Hủy", hệ thống sẽ bỏ qua thao tác xóa và hiển thị danh sách danh mục trong bảng DanhMuc**.**
    2. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case, nếu không kết nối được với cơ sở dữ liệu, hệ thống sẽ hiển thị một thông báo lỗi "Lỗi kết nối" và use case kết thúc.
* **Các yêu cầu đặc biệt:** Use case này chỉ cho phép người quản lý thực hiện.
* **Tiền điều kiện:** Người quản lý cần đăng nhập với vai trò quản lý trướckhi có thể thực hiện use case**.**
* **Hậu điều kiện:** Nếu use case kết thúc thành công, thông tin về danh mục món ăn sẽ được cập nhật vào CSDL.
* **Điểm mở rộng:** Không có.

**2.3.7 Mô tả Use Case Quản lý món ăn**

* **Tên Use Case:** Quản lý món ăn
* **Mô tả vắn tắt**: Người quản lý có thể thêm, sửa, xóa thông tin món ăn và cập nhật trạng thái.
* **Luồng các sự kiện** 
  + **Luồng cơ bản:** 
    1. Use case bắt đầu khi người quản lý chọn chức năng "Quản lý món ăn" trên menu quản trị. Hệ thống lấy tất cả thông tin về món ăn từ bảng MonAn trong cơ sở dữ liệu và hiển thị danh sách món ăn theo danh mục lên màn hình.
    2. **Thêm thông tin món ăn:**

**a.** Người quản lý kích vào nút "Thêm mới" trên cửa sổ quản lý món ăn. Hệ thống hiển thị màn hình yêu cầu nhập thông tin về món ăn.

**b.** Người quản lý nhập đầy đủ các thông tin (Tên món ăn, Mô tả, Giá, chọn Danh mục) và kích vào nút "Tạo". Hệ thống sẽ tạo một MaMonAn mới, cập nhật bảng MonAn và hiển thị danh sách món ăn đã được cập nhật.

* + 1. **Sửa thông tin món ăn:**

a.Người quản lý kích vào nút "Sửa" trên một hàng món ăn. Hệ thống lấy tất cả thông tin cũ của món ăn được chọn**.**

b.Người quản lý nhập đầy đủ thông tin mới cho món ăn đó sau đó kích vào nút "Cập nhật". Hệ thống sẽ cập nhật thông tin của món ăn được chọn trong bảng MonAn và hiển thị danh sách món ăn đã được cập nhật.

* + 1. **Xóa thông tin món ăn:**

**a.** Người quản lý kích vào nút "Xóa" trên một dòng món ăn. Hệ thống sẽ hiển thị một màn hình yêu cầu xác nhận xóa.

**b.** Người quản lý kích vào nút "Đồng ý". Hệ thống sẽ xóa món ăn được chọn khỏi bảng MonAn và hiển thị danh sách món ăn đã được cập nhật.

* + **Luồng rẽ nhánh:** 
    1. Tại bước 4b trong luồng cơ bản, nếu người quản lý kích vào nút "Hủy", hệ thống sẽ bỏ qua thao tác xóa và hiển thị danh sách món ăn trong bảng MonAn.
    2. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case, nếu không kết nối được với cơ sở dữ liệu, hệ thống sẽ hiển thị một thông báo lỗi "Lỗi kết nối" và use case kết thúc.
* **Các yêu cầu đặc biệt:** Use case này chỉ cho phép người quản lý thực hiện.
* **Tiền điều kiện:** Người quản lý cần đăng nhập với vai trò quản lý trước khi có thể thực hiện use case.
* **Hậu điều kiện:** Nếu use case kết thúc thành công, thông tin về món ăn sẽ được cập nhật vào CSDL.
* **Điểm mở rộng:** Không có.

**2.3.8 Mô tả Use Case Quản lý nhân viên**

* **Tên Use Case:** Quản lý nhân viên
* **Mô tả vắn tắt:** Người quản lý có thể thêm, sửa, xóa thông tin nhân viên.
* **Luồng các sự kiện** 
  + **Luồng cơ bản:** 
    1. Use case bắt đầu khi người quản lý chọn chức năng "Quản lý nhân viên" trên menu quản trị. Hệ thống lấy tất cả thông tin về nhân viên từ bảng NhanVien trong cơ sở dữ liệu và hiển thị danh sách nhân viên lên màn hình.
    2. **Thêm thông tin nhân viên:**

**a.** Người quản lý kích vào nút "Thêm mới" trên cửa sổ quản lý nhân viên. Hệ thống hiển thị màn hình yêu cầu nhập thông tin về nhân viên.

**b.** Người quản lý nhập đầy đủ các thông tin (Tên NV, Tài khoản, Mật khẩu, Vai trò) và kích vào nút "Thêm". Hệ thống sẽ tạo một MaNV mới, cập nhật bảng NhanVien và hiển thị danh sách nhân viên đã được cập nhật.

* + 1. **Sửa thông tin nhân viên:**

**a.** Người quản lý kích vào nút "Sửa" trên một hàng nhân viên. Hệ thống lấy tất cả thông tin cũ của nhân viên được chọn.

**b.** Người quản lý nhập đầy đủ thông tin mới cho nhân viên đó sau đó kích vào nút "Cập nhật". Hệ thống sẽ cập nhật thông tin của nhân viên được chọn trong bảng NhanVien và hiển thị danh sách nhân viên đã được cập nhật.

* + 1. **Xóa thông tin nhân viên:**

**a.** Người quản lý kích vào nút "Xóa" trên một dòng nhân viên. Hệ thống sẽ hiển thị một màn hình yêu cầu xác nhận xóa.

**b.** Người quản lý kích vào nút "Đồng ý". Hệ thống sẽ xóa nhân viên được chọn khỏi bảng NhanVien và hiển thị danh sách nhân viên đã được cập nhật.

* + **Luồng rẽ nhánh:** 
    1. Tại bước 4b trong luồng cơ bản, nếu người quản lý kích vào nút "Hủy", hệ thống sẽ bỏ qua thao tác xóa và hiển thị danh sách nhân viên trong bảng NhanVien.
    2. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case, nếu không kết nối được với cơ sở dữ liệu, hệ thống sẽ hiển thị một thông báo lỗi "Lỗi kết nối" và use case kết thúc.
* **Các yêu cầu đặc biệt:** Use case này chỉ cho phép người quản lý thực hiện.
* **Tiền điều kiện:** Người quản lý cần đăng nhập với vai trò quản lý trước khi có thể thực hiện use case.
* **Hậu điều kiện:** Nếu use case kết thúc thành công, thông tin về nhân viên sẽ được cập nhật vào CSDL**.**
* **Điểm mở rộng:** Không có.

**2.3.9 Mô tả Use Case Thống kê**

* **Tên Use Case:** Thống kê
* **Mô tả vắn tắt:** Use case này cho phép quản lý xem các số liệu thống kê.
  + **Luồng cơ bản:**
    1. Use case này bắt đầu khi quản lý chọn chức năng "Thống kê". Hệ thống hiển thị các tùy chọn thống kê (doanh thu, món ăn).
    2. Quản lý chọn loại thống kê và khoảng thời gian (nếu cần). Hệ thống hiển thị kết quả thống kê. Use case kết thúc.
  + **Luồng rẽ nhánh:**
    1. Nếu không có dữ liệu trong khoảng thời gian đã chọn, hệ thống hiển thị thông báo.
* **Các yêu cầu đặc biệt:** Không có.
* **Tiền điều kiện:** Quản lý đã đăng nhập vào hệ thống.
* **Hậu điều kiện:** Dữ liệu thống kê được hiển thị.
* **Điểm mở rộng:** Không có.

**2.3.10 Mô tả Use Case Quản lý hóa đơn**

**Tên Use Case:** Quản lý hóa đơn sau thanh toán

* **Mô tả vắn tắt:** Use case này cho phép quản lý xem lại các hóa đơn đã thanh toán.
  + **Luồng cơ bản:**
    1. Use case này bắt đầu khi quản lý chọn chức năng "Quản lý hóa đơn". Hệ thống hiển thị danh sách hóa đơn đã thanh toán.
    2. Quản lý có thể xem chi tiết từng hóa đơn. Use case kết thúc.
  + **Luồng rẽ nhánh:**
    1. Quản lý có thể lọc hóa đơn theo khoảng thời gian (tùy chọn).
* **Các yêu cầu đặc biệt:** Không có.
* **Tiền điều kiện:** Quản lý đã đăng nhập vào hệ thống.
* **Hậu điều kiện:** Danh sách hóa đơn đã thanh toán được hiển thị.
* **Điểm mở rộng:** Không có.

**2.3.11 Use case Quản lí danh sách bàn**

* **Tên Use Case:** Quản lý bàn
* **Mô tả vắn tắt:** Người quản lý hoặc người quản trị có thể thêm, sửa, xóa thông tin các bàn trong nhà hàng.
* **Luồng các sự kiện** 
  + **Luồng cơ bản:** 
    1. Use case bắt đầu khi người quản lý chọn chức năng "Quản lý bàn" trên menu quản trị. Hệ thống lấy tất cả thông tin về bàn (mã bàn, tên bàn, số chỗ) từ bảng Ban trong cơ sở dữ liệu và hiển thị danh sách bàn lên màn hình.
    2. **Thêm thông tin bàn:**

**a.** Người quản lý kích vào nút "Thêm mới" trên cửa sổ quản lý bàn. Hệ thống hiển thị màn hình yêu cầu nhập thông tin về bàn.

**b.** Người quản lý nhập đầy đủ các thông tin (ví dụ: Tên bàn/Số bàn, Số chỗ ngồi).

**c.** Người quản lý kích vào nút "Tạo". Hệ thống sẽ tạo một MaBan mới, cập nhật bảng Ban và hiển thị danh sách bàn đã được cập nhật.

* + 1. **Sửa thông tin bàn:**

**a.** Người quản lý kích vào nút "Sửa" trên một hàng bàn trong danh sách. Hệ thống lấy tất cả thông tin cũ của bàn được chọn và hiển thị trên màn hình chỉnh sửa.

**b.** Người quản lý nhập đầy đủ thông tin mới cho bàn đó (ví dụ: thay đổi Tên bàn/Số bàn, Số chỗ tối đa).

**c.** Người quản lý kích vào nút "Cập nhật". Hệ thống sẽ cập nhật thông tin của bàn được chọn trong bảng Ban và hiển thị danh sách bàn đã được cập nhật.

* + 1. **Xóa thông tin bàn:**

**a.** Người quản lý kích vào nút "Xóa" trên một dòng bàn trong danh sách. Hệ thống sẽ hiển thị một màn hình yêu cầu xác nhận xóa (ví dụ: "Bạn có chắc chắn muốn xóa bàn [Tên bàn] không?").

**b**. Người quản lý kích vào nút "Đồng ý". Hệ thống sẽ xóa bàn được chọn khỏi bảng Ban và hiển thị danh sách bàn đã được cập nhật.

* + **Luồng rẽ nhánh:** 
    1. Tại bước 4b trong luồng cơ bản, nếu người quản lý kích vào nút "Hủy", hệ thống sẽ bỏ qua thao tác xóa và hiển thị danh sách bàn trong bảng Ban như ban đầu.
    2. Tại bất kỳ thời điểm nào trong quá trình thực hiện use case, nếu không kết nối được với cơ sở dữ liệu, hệ thống sẽ hiển thị một thông báo lỗi "Lỗi kết nối" và use case kết thúc**.**
* **Các yêu cầu đặc biệt:** Use case này chỉ cho phép người quản lý thực hiện.
* **Tiền điều kiện:** Người quản lý cần đăng nhập với vai trò quản lý trước khi có thể thực hiện use case.
* **Hậu điều kiện:** Nếu use case kết thúc thành công, thông tin về bàn sẽ được thêm, sửa hoặc xóa trong CSDL.
* **Điểm mở rộng:** Không có.

**2.3.12 Use case Quản lý hoá đơn**

* **Tên Use Case:** Quản lý hóa đơn
* **Mô tả vắn tắt:** Use case này cho phép người quản lý xem danh sách tất cả hóa đơn (bao gồm cả đã thanh toán và chưa thanh toán), xem chi tiết từng hóa đơn, và xác nhận trạng thái "Đã thanh toán" cho các hóa đơn chưa thanh toán.
* **Luồng các sự kiện:** 
  + **Luồng cơ bản:** 
    1. Use case bắt đầu khi người quản lý chọn chức năng "Quản lý hóa đơn" từ menu quản trị.
    2. Hệ thống lấy tất cả thông tin hóa đơn từ cơ sở dữ liệu bao gồm cả những hóa đơn đã thanh toán và chưa thanh toán và hiển thị.
    3. Người quản lý kích chọn một hóa đơn cụ thể từ danh sách.
    4. Hệ thống chuyển sang một trang mới, hiển thị thông tin chi tiết của hóa đơn được chọn (gồm thông tin bàn, nhân viên phục vụ, danh sách món ăn, số lượng, giá, ghi chú, tổng tiền, phương thức thanh toán, và trạng thái hiện tại).
    5. Xử lý hóa đơn chưa thanh toán (thanh toán chuyển khoản):

**a.** Nếu Trạng thái hóa đơn là "Chưa thanh toán" (và phương thức thanh toán thường là chuyển khoản): Hệ thống hiển thị nút "Xác nhận đã thanh toán".

**b.** Khi người quản lý nhận được thông báo xác nhận đã nhận tiền từ ngân hàng (ngoài hệ thống), người quản lý kích vào nút "Xác nhận đã thanh toán".

**c.** Hệ thống cập nhật trạng thái của hóa đơn đó trong bảng HoaDon sang "Đã thanh toán".

**d.** Hệ thống hiển thị thông báo xác nhận và cập nhật lại thông tin chi tiết hóa đơn, nút "Xác nhận đã thanh toán" không còn hiển thị. Use case kết thúc.

* + 1. Xử lý hóa đơn đã thanh toán : Hệ thống chỉ thông tin chi tiết của hóa đơn. Nút "Xác nhận đã thanh toán" sẽ không được hiển thị vì hóa đơn đã hoàn tất thanh toán. Use case kết thúc.
  + **Luồng rẽ nhánh:** 
    1. Lỗi kết nối/Truy xuất dữ liệu: Tại bất kỳ thời điểm nào trong quá trình thực hiện use case, nếu không kết nối được với backend hoặc cơ sở dữ liệu, hệ thống sẽ hiển thị một thông báo lỗi "Lỗi kết nối hệ thống. Vui lòng thử lại sau." và use case kết thúc.
* **Các yêu cầu đặc biệt:** Use case này chỉ cho phép người quản lý hoặc người quản trị thực hiện.
* **Tiền điều kiện:** Người quản lý (hoặc quản trị) cần đăng nhập với vai trò quản lý trước khi có thể thực hiện use case.
* **Hậu điều kiện:** Nếu use case kết thúc thành công, trạng thái của hóa đơn được chọn (nếu là "Chưa thanh toán") sẽ được cập nhật thành "Đã thanh toán" trong cơ sở dữ liệu.
* **Điểm mở rộng:** Không có.

**2.4 Biểu đồ trình tự**

**2.4.1 Use case Dang Nhap**

**A diagram of a diagram

AI-generated content may be incorrect.**

**2.4.2 Use case Quản lý bàn(Nhân viên)**

**A diagram of a diagram

AI-generated content may be incorrect.**

**2.4.3 Use case Order Món Ăn**

**A diagram of a project

AI-generated content may be incorrect.**

**2.4.4 Use case Tạo hoá đơn**

**A diagram of a diagram

AI-generated content may be incorrect.**

**2.4.5 Use case Đăng ký nhà hàng**

**A diagram of a diagram

AI-generated content may be incorrect.**

**2.4.6 Use case Quản lý danh mục**

**A diagram of a project

AI-generated content may be incorrect.**

**A diagram of a project

AI-generated content may be incorrect.**

**2.4.7 Use case Quản lý món ăn**

**A diagram of a company

AI-generated content may be incorrect.**

**2.4.8 Use case Quản lý nhân viên**

**A diagram of a company

AI-generated content may be incorrect.**

**2.4.11 Use case Quản lý danh sách bàn**

**A diagram of a diagram

AI-generated content may be incorrect.**