Group Project Documentation:   
Advanced Cross-Platform Application Programming with NET

*Laundry Middle Platform*

**Prepared by Group 2**

***Lương Thế Dân***

***Nguyễn Hữu Toàn***

***Trần Bảo Sơn***

**Ho Chi Minh City, 2023**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1.** **Project Introduction 1**

1.1 Product Perspective 1

1.2 User Classes and Characteristics 1

**2.** **Database Design 1**

**3.** **System Architecture 1**

**4.** **Implementation 2**

4.1. Deployment Considerations [2](#_heading=h.2s8eyo1)

4.2. Screenshots and explanations [2](#_heading=h.17dp8vu)

**5.** **References 2**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| *Dan* | *9/23/2023* | *Initial draft* | *1.0 draft 1* |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Project Introduction

## Product Perspective

In response to the increasing demand for convenient laundry services, our company is developing a laundry service management system. This system is designed to streamline the process of creating, managing, and delivering laundry orders for customers. The primary objective is to provide a hassle-free and efficient platform for individuals seeking laundry and drying services from various stores.

## User Classes and Characteristics

**Main actor:**

* Customer
* Admin
* Laundry store
* Staff

**Functional Requirement:**

**Admin:**

1. **Manage Stores:**

* The admin should be able to add, edit, or remove laundry stores from the system. This includes updating store information, such as location and contact details.

1. **Monitor Orders:**

* The admin should have access to a dashboard where they can view and track all customer orders and their statuses.

1. **User Management:**

* The admin should have the ability to manage customer accounts, including creating, editing, and deactivating accounts.

**Customer:**

1. **Create Order:**

* Customers should be able to create new laundry orders, specifying the weight of their items and the preferred laundry time (express, quick, regular, etc.).

1. **Select Store:**

* Customers should choose a laundry store from the provided list when placing an order.

1. **View Order Status:**

* Customers should be able to check the status of their orders in real-time, including whether they're
* in process, completed, or out for delivery.

1. **Make Payments:**

* Customers should have a payment option to settle the bill for the laundry service.

1. **View Store Information:**

* Customers should access information about the laundry stores, including pricing and service descriptions.

**Staff:**

1. **Receive Orders:**

* Staff members should have the capability to receive new orders placed by customers.

1. **Deliver Orders:**

* Staff members should assign received orders to specific laundry stores for processing.

1. **Update Order Status:**

* Staff members should update the status of orders as they progress through the system, from pickup to delivery.

**Laundry Store:**

1. **Receive Orders:**

* Laundry stores should receive orders from the system, including details about the laundry items and customer preferences.

1. **Process Orders:**

* Stores should perform laundry and drying services according to the order specifications.

1. **Update Order Status:**

* After completing the laundry, stores should update the order status to indicate that it's ready for pickup.

1. **Manage Store Information:**

* Stores should be able to update their service descriptions, pricing, and contact details within the system.

**Business Rule**

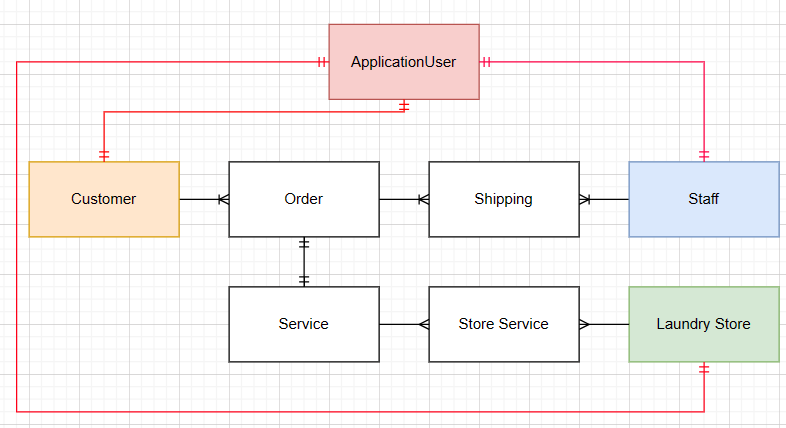
| **Rule ID** | **Rule Title** | **Function** | **Rule Description** |
| --- | --- | --- | --- |
| BR-O-1 | Order date | Order | Order date equals time when pressing submit order |
| BR-O-2 | ExpectedFinishDate | Order | - Equal to: OrderDate + 1 Hour + 1 Hour \* Total unfinished order of laundry store |
| BR-O-3 | Address | Order | Order address equal to customer’s address |
| BR-O-4 | Total Price | Order | Total price equals amount \* service price |
|  |  |  |  |
| BR-O-C-1 | Enter amounts | Order | - Required  only accepts Numeric Characters  - Cannot be bigger than 20; |
| BR-O-C-2 | Choose time frame | Order | required  only accepts Numeric Characters  Range [1, 2, 3] |
| BR-O-C-3 | Choose order type | Order | - Required  - Must be chosen from provided order type option (oneway, twoway) |
|  |  |  |  |
| BR-OH-C-1 | View status | View | Customers can view their orders' final status here. |
|  |  |  |  |
| BR-S-1 | Type | UpdateSStatus | -Required  Staff must choose type (Collect from customer, delivery to customer) |
| BR-S-S-1 | Update staff order status | UpdateSStatus | Staff must update the status when they deliver stuff to the laundry store or to customers. |
|  |  |  |  |
| BR-LSS-LS-1 | Name | AddService | * Required * LS must write legal name for the service |
| BR-LSS-LS-2 | Description | AddService | * Required * LS must write legal description with a specific name for the service |
| BR-LSS-LS-3 | Price | AddService | * Required * LS must write legal name for the service |

**Business Process/Workflow**

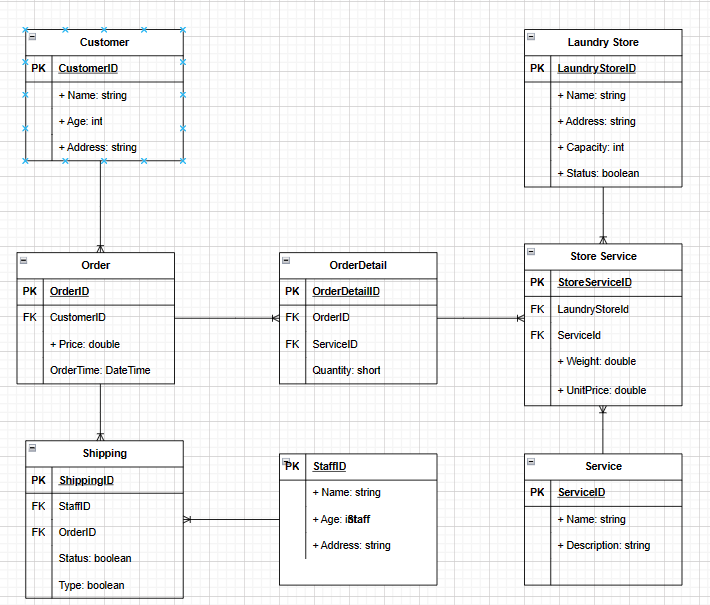
1. Admin → Add Store → View Details → Edit/Remove Store as Needed
2. Admin -> Add Staff -> View Details -> Edit or remove staff as needed. Select Store → Select Service → Place Order → Update Order Status ("Pending") → update order list for delivery.
3. Platform Staff → View OrderType→ View DeliverStatus ("In Process") → Deliver Order → Update DeliverStatus ("Finished") → update order list for laundry store
4. Laundry Store → Receive Orders → Update LaundryStatus ("In Process") → Process Order → Update LaundryStatus ("Finished") → Update Order list for delivery

# Database Design

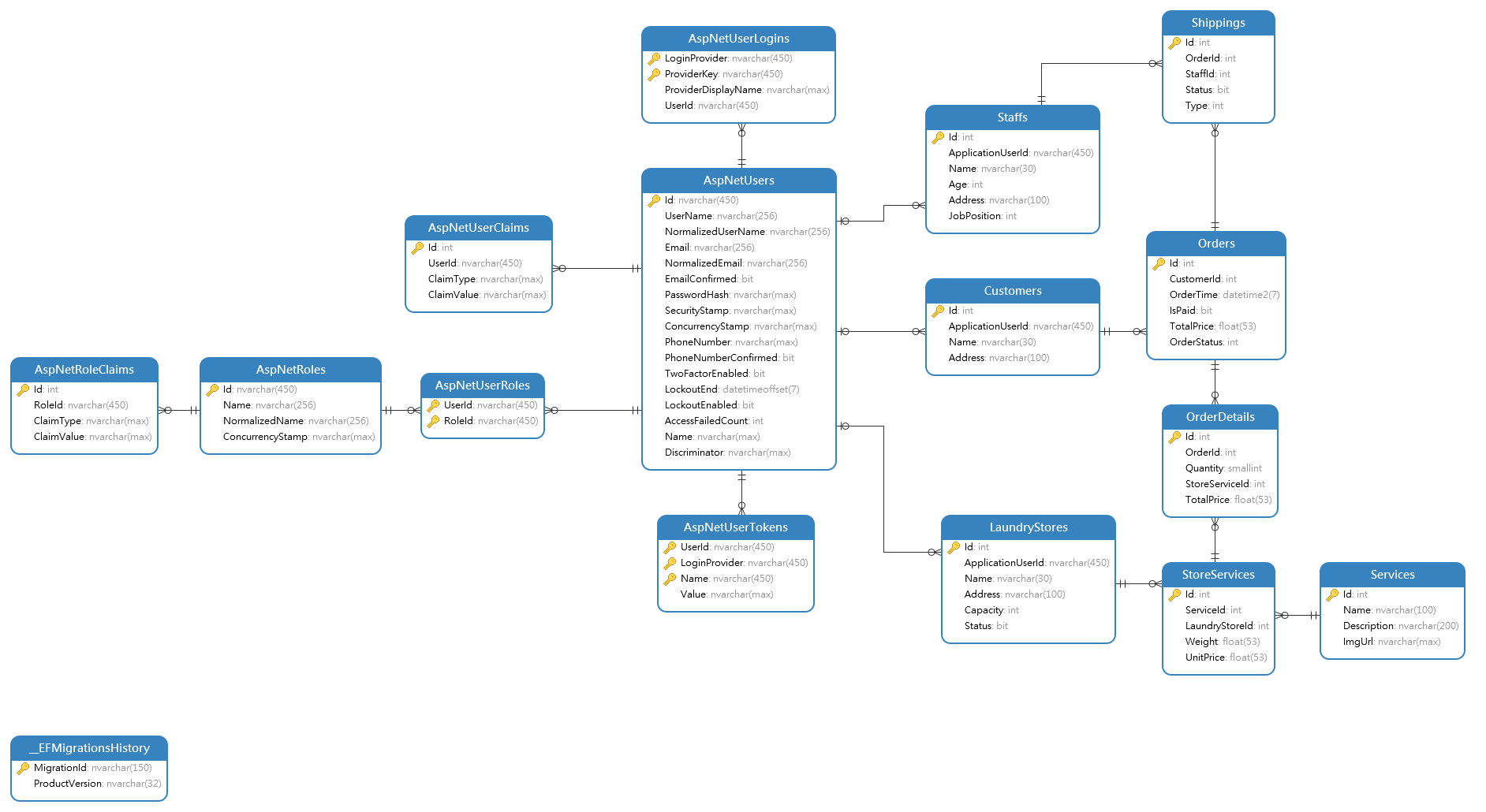
**ERD:**

****

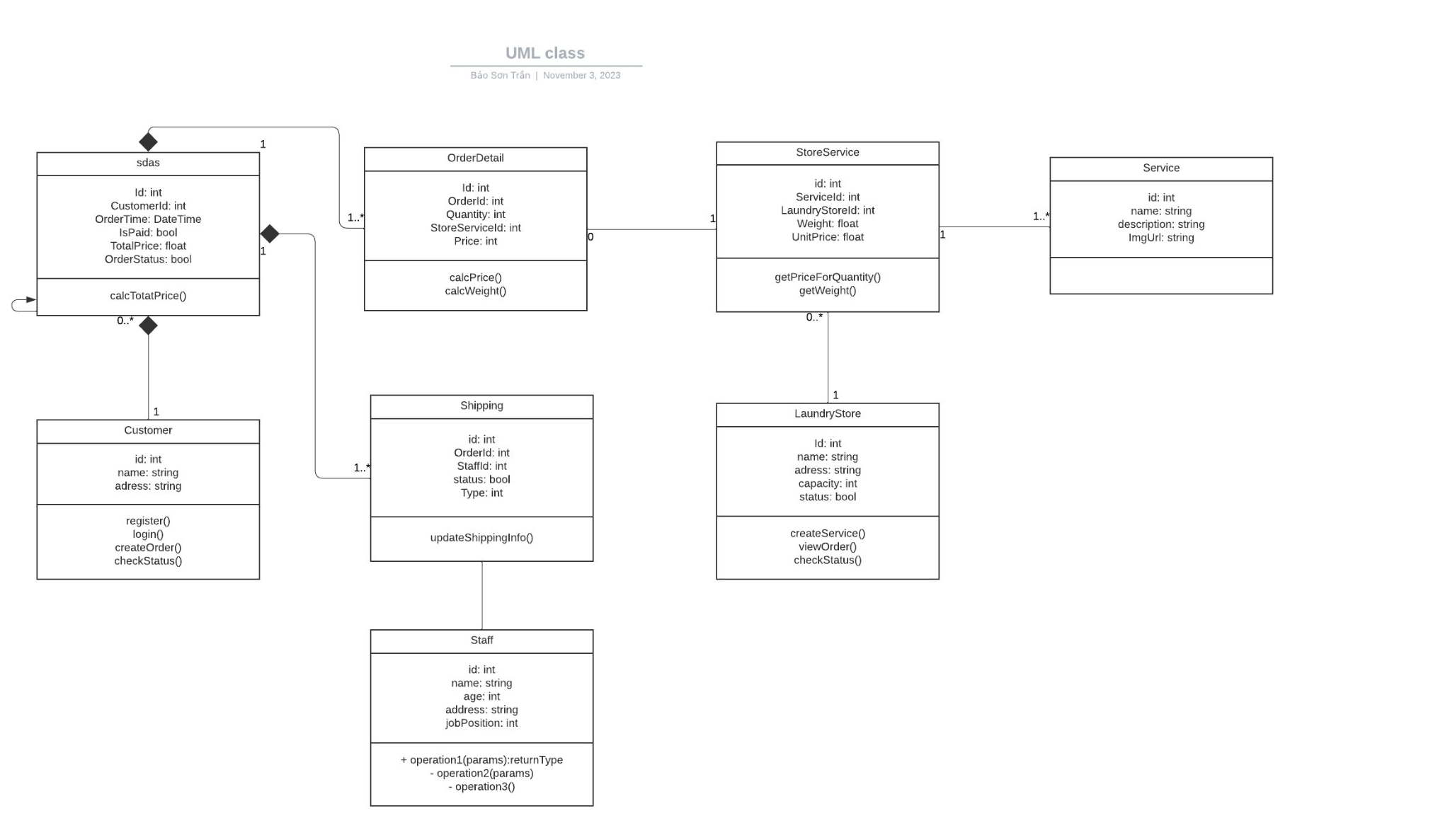
**ERD Logical:**

****

**ERD Physical:**

****

# Class Design



# System Architecture

These are the technologies that will be used in the making of this project:

· Database: MS SQL

· ORM: Microsoft Entity Framework Core

· Framework: Microsoft .NET

· View: Razor Page

· 3rd Party API: Google SMTP, Bootstrap 5, Datatables

For the design pattern, we will be using the generic repository architecture with a four-layer structure.

# Implementation

## Deployment Considerations

System Requirements

| PC | Minimum | Recommend |
| --- | --- | --- |
| Internet Connection | LAN, Internet Access capability | LAN, Wifi (16Mbps) |
| Processor | Intel Core i5 1.5Ghz | Intel Core i7 2.6Ghz |
| Memory | 4GB RAM | 8GB RAM and above |
| Storage | 4GB | 6GB up |
| Web Browser | Chrome (v69) | Chrome latest stable version |

Software requirement

| Component | Name and version | Description |
| --- | --- | --- |
| Operating System | Window 10/11, other open sources (Linux, Ubuntu) |  |
| Database | SqlServer 19 |  |
| .NET Core Runtime | 5.0 | Used to hosting API Server |

## Screenshots and explanations

*<Screen flow | Dialog Map>*

*<The screenshots and explanations>*

# References

*N/A*