CAPSTONE PROJECT REPORT

Report 4 – Software Design Document

Table of Contents

I. Record of Changes

Date	A* M, D	In charge	Change Description
2/6/2023	A	HungLM	System design, Detailed design
2/6/2023	A	QuocTK	Database Design
5/6/2023	M	HungLM	Database Design, Detailed Design
5/6/2023	M	QuocTK	Database Design

*A - Added M - Modified D - Deleted

II. Software Design Document

1. System Design

1.1 System Architecture

1.1.1 Overall Architecture Design
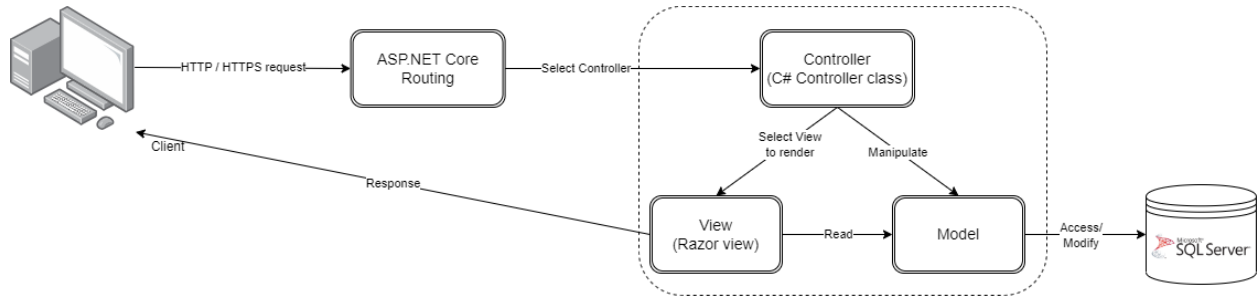
Figure 1.1.1.1: Overall Architecture Design

1.2 Package Diagram
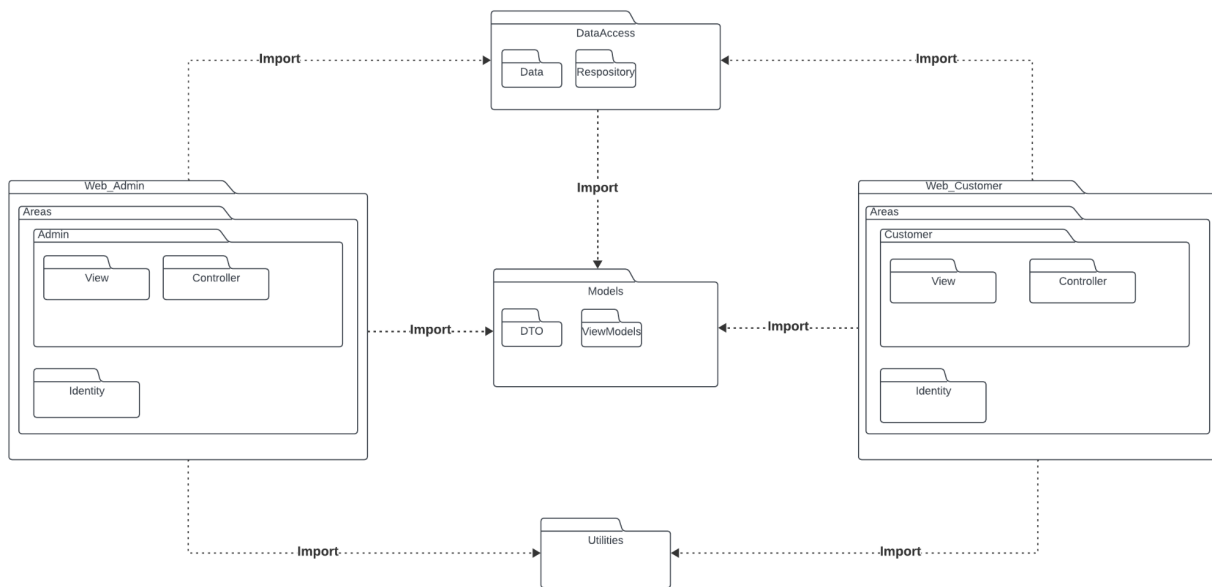
Figure 1.2.1: Package Diagram

Package Descriptions

No	Package	Description
01	Data	Responsible for managing the connection, mapping models to tables, change tracking, executing queries, and managing transactions when interacting with a database in application.
02	Models	Representation class of entity in database
03	DTO	Data transfer object
04	ViewModels	Objects only to view purpose
05	Controllers	System controllers
06	Identity	Identity configure
07	Views	Web UI
08	Utilities	Contains Helper Class such as: StaticDetail, EmailSender,...

2. Database Design

Figure 2.1: Database Diagram

Table Descriptions

2.1 Tables

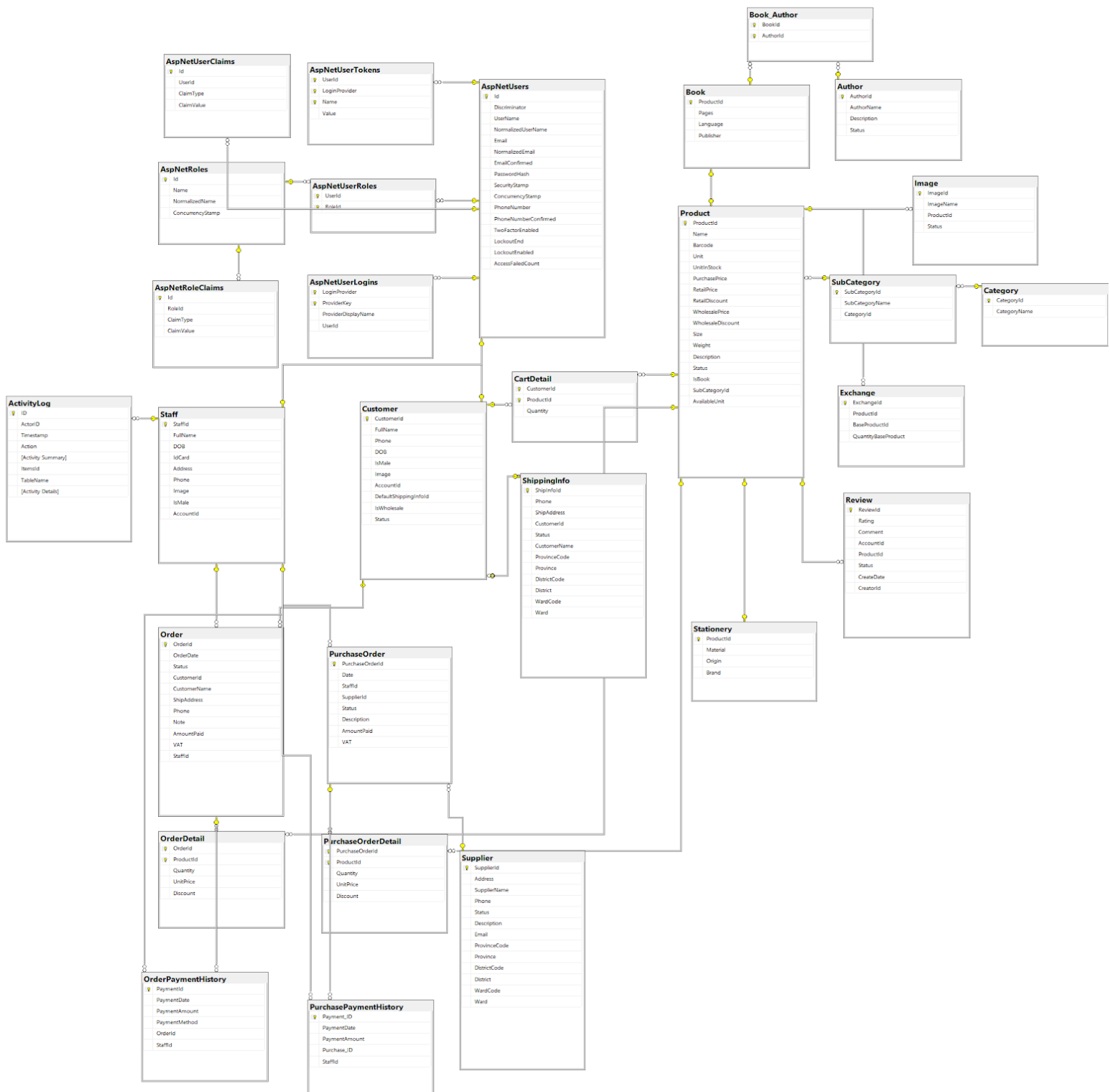| No	Table	Description
1	AspNetUserClaims	Store additional claims or user-specific information associated with a user
2	AspNetRoles	Store user roles or groups that define access permissions and behaviors within the application.
3	AspNetRoleClaims	Store additional claims or role-specific information associated with user roles. These claims can define custom permissions or attributes for specific roles
4	AspNetUserTokens	Store authentication tokens associated with users, which can be used for various security-related operations, such as email confirmation, password reset, and two-factor authentication
5	AspNetUserRoles	Associate users with their respective roles. It defines the many-to-many relationship between users and roles, indicating which roles a user belongs to within the application.
6	AspNetUserLogins	Store external login information for users, such as logins through external identity providers like Google, Facebook, or Microsoft. It allows users to associate multiple external accounts with their application account.
7	AspNetUsers	Store user account information, including authentication credentials (such as hashed passwords), user properties, and related data. It forms the core table for managing user identities within the application.
8	ActivityLog	Store list of system activities
9	Staff	Store list of staffs
10	Customer	Store list of customers
11	Order	Store list of orders
12	OrderDetail	Store list of order details
13	OrderPaymentHistory	Store list of order payments
14	PurchaseOrder	Store list of purchase orders
15	PurchaseOrderDetail	Store list of purchase order details
16	PurchasePaymentHistory	Store list of purchase order payments
17	Supplier	Store list of suppliers
18	ShippingInfo	Store list of customer's shipping information
19	CartDetail	Store list of customer's cart details
20	Product	Store list of products
21	Book	Store list of product's subtype books' additional information
22	Author	Store list of book authors
23	Book_Author	Store many-many relationship of books and authors
24	Stationery	Store list of product's subtype stationeries' additional information
25	Image	Store list of product's images
26	Category	Store list of product categories
27	SubCategory	Store list of product sub-categories
28	Exchange	Store list of exchanges between products (for product version control)

29	Review		Store list of product reviews

2.2 Attributes

2.2.1 AspNet User Claims table

No	Field Name	Type	Description
1	Id	int	Primary Key Auto increment Unique claim identifier
2	UserId	nvarchar(450)	Foreign key to AspNetUsers
3	ClaimType	nvarchar(MAX)	Represents the type of the claim
4	ClaimValue	nvarchar(MAX)	The value associated with the claim type

2.2.2 AspNet Roles table

No	Field Name	Type	Description
1	Id	nvarchar(450)	Primary Key A unique identifier for each role
2	Name	nvarchar(256)	Name of the role
3	NormalizedName	nvarchar(256)	Normalized name of the role (usually uppercase)
4	Concurrency Stamp	nvarchar(MAX)	Concurrency stamp for optimistic concurrency

2.2.3 AspNet Role Claims table

No	Field Name	Type	Description
1	Id	int	Primary Key Auto increment Unique claim identifier
2	RoleId	nvarchar(450)	Foreign key referencing the associated role
3	ClaimType	nvarchar(MAX)	Type of the claim
4	ClaimValue	nvarchar(MAX)	Value of the claim

2.2.4 AspNet User Tokens table

No	Field Name	Type	Description
1	UserId	nvarchar(450)	Primary Key Foreign key referencing the associated user
2	LoginProvider	nvarchar(450)	Primary Key The external login provider (e.g., Google)
3	Name	nvarchar(450)	Primary Key The name of the token (e.g., "authenticator")
4	Value	nvarchar(MAX)	The token value

2.2.5 AspNet Role Claims table

No	Field Name	Type	Description

1	Id	int	Primary Key Auto increment Unique identifier for the role claim.
2	RoleId	nvarchar(450)	Foreign key reference to the "Id" field in the "AspNetRoles" table, indicating the associated role.
3	ClaimType	nvarchar(MAX)	Represents the type of the claim being stored.
4	ClaimValue	nvarchar(MAX)	Holds the value of the claim.

2.2.6 AspNet User Roles table

No	Field Name	Type	Description
1	UserId	nvarchar(450)	Foreign key reference to the "Id" field in the "AspNetUsers" table, indicating the associated user.
2	RoleId	nvarchar(450)	Foreign key reference to the "Id" field in the "AspNetRoles" table, indicating the associated role.

2.2.7 AspNet User Logins table

No	Field Name	Type	Description
1	LoginProvider	nvarchar(450)	The external login provider, e.g., "Google", "Facebook".
2	ProviderKey	nvarchar(450)	The unique identifier provided by the external provider.
3	ProviderDisplayName	nvarchar(MAX)	A user-friendly display name for the external provider.
4	UserId	nvarchar(450)	Foreign key reference to the user's primary key.

2.2.8 AspNetUsers table

No	Field Name	Type	Description
1	Id	nvarchar(450)	Primary Key Unique identifier for the user.
2	UserName	nvarchar(256)	The username used for authentication.
3	NormalizedUserName	nvarchar(256)	The normalized version of the username.
4	Email	nvarchar(256)	The user's email address.
5	NormalizedEmail	nvarchar(256)	The normalized version of the email address.
6	EmailConfirmed	bit	Indicates if the user's email is confirmed.
7	PasswordHash	nvarchar(MAX)	The hashed password for the user.
8	SecurityStamp	nvarchar(MAX)	A random value used to improve security.
9	ConcurrencyStamp	nvarchar(MAX)	Used for concurrency control.
10	PhoneNumber	nvarchar(MAX)	The user's phone number.
11	PhoneNumberConfirmed	bit	Indicates if the user's phone number is confirmed.
12	TwoFactorEnabled	bit	Indicates if two-factor authentication is enabled.
13	LockoutEnd	datetimeoffset(7)	Date and time when lockout ends (if applicable).
14	LockoutEnabled	bit	Indicates if user lockout is enabled

No	Field Name	Type	Description
15	AccessFailedCount	int	Number of failed access attempts.

2.2.9 Activity Log table

No	Field Name	Type	Description
1	ID	int	ID of an activity Primary Key Auto increment
2	ActorID	int	ID of the staff doing the activity Foreign key reference to "StaffId" of "Staff" table
3	Timestamp	datetime	Time of the activity
4	Action	nvarchar(30)	Type of activity
5	Activity Summary	nvarchar(100)	Short summary of the activity
6	ItemsId	nvarchar(30)	ID of the item affected by the activity
7	TableName	nvarchar(50)	Name of the table affected by the activity
8	Activity Details	ntext	Detail description of the activity

2.2.10 Staff table

No	Field Name	Type	Description
1	StaffId	int	ID of staff Primary Key Auto increment
2	FullName	nvarchar(70)	Full name of staff
3	DOB	date	Birthdate of staff
4	IdCard	nchar(12)	National Id card number of staff
5	Address	nvarchar(12)	Home address of staff
6	Phone	nchar(10)	Phone number of staff
7	Image	nvarchar(150)	Directory to the avatar image of staff
8	IsMale	bit	Staff gender True is Male False is Female
9	AccountId	nvarchar(450)	Account ID of staff Foreign key reference to "ID" of "AspNetUsers" table Unique

2.2.11 Customer table

No	Field Name	Type	Description
1	CustomerId	int	ID of customer Primary Key Auto increment
2	FullName	nvarchar(100)	Full name of customer
3	Phone	nchar(10)	Phone number of customer
4	DOB	date	Birthdate of customer
5	IsMale	bit	Gender of customer True for Male False for Female
6	Image	nvarchar(150)	Directory to the avatar image of customer
7	AccountId	nvarchar(450)	Account ID of customer Foreign key reference to "ID" of "AspNetUsers" table Unique
8	Default ShippingInfo	int	Customer's default shipping information ID Foreign key reference to "ShipInfoId" of "ShippingInfo" table
9	IsWholesale	bit	Customer type - retail or wholesale True for Wholesale customer False for Retail customer
10	Status	bit	Customer's status True for Active False for Inactive

2.2.12 Order table

No	Field Name	Type	Description
1	OrderId	int	ID of Order Primary Key Auto increment
2	OrderDate	datetime	Order's creation date
3	Status	nvarchar(50)	Status of order
4	CustomerId	int	ID of order's customer Foreign key reference to "CustomerID" of "Customer" table
5	Customer Name	nvarchar(100)	Recipient name of order
6	ShipAddress	nvarchar(300)	Shipping Address of order
7	Phone	nchar(10)	Recipient's phone number of order
8	Note	nvarchar(MAX)	Order's note
9	AmountPaid	int	Amount of money already paid for the order
10	VAT	float	Order's VAT
11	StaffId	int	ID of the staff who handles order Foreign key reference to "StaffId" of "Staff" table

2.2.13 Order Detail table

No	Field Name	Type	Description
1	OrderId	int	ID of order this detail belongs to Primary Key Foreign key reference to "OrderId" of "Order" table
2	ProductId	int	ID of product bought Primary key Foreign key reference to "ProductId" of "Product" table
3	Quantity	int	The number of products bought
4	UnitPrice	int	The price of 1 unit of product bought
5	Discount	float	Discount of the product bought

2.2.14 Order Payment History table

No	Field Name	Type	Description
1	PaymentId	int	ID of the order payment Primary Key Auto increment
2	Payment Date	datetime	Date of the order payment
3	Payment Amount	decimal(15, 0)	The amount of money paid
4	Payment Method	nvarchar(MAX)	The method of payment
5	OrderId	int	ID of the order being paid Foreign key reference to "OrderId" of "Order" table
6	StaffId	int	ID of the staff handling the payment Foreign key reference to "StaffId" of "Staff" table

2.2.15 Purchase Order table

No	Field Name	Type	Description
1	Purchase OrderId	int	ID of the purchase order Primary Key Auto increment
2	Date	datetime	Purchase order's creation date
3	StaffId	int	ID of the staff handling the purchase order Foreign key reference to "StaffId" of "Staff" table
4	SupplierId	int	ID of the Supplier purchased from Foreign key reference to "SupplierId" of "Supplier" table
5	Status	nvarchar(30)	Purchase order's status
6	Description	nvarchar(MAX)	Purchase order's optional description
7	AmountPaid	decimal(15, 0)	Amount of money already paid for the purchase order
8	VAT	float	Purchase order's VAT

2.2.16 Purchase Order Detail table

No	Field Name	Type	Description
1	Purchase OrderId	int	ID of the purchase order this detail belongs to Primary Key Foreign key reference to "PurchaseOrderId" of "PurchaseOrder" table
2	ProductId	int	ID of product bought Primary key Foreign key reference to "ProductId" of "Product" table
3	Quantity	int	The number of products bought
4	UnitPrice	decimal(7, 0)	The price of 1 unit of product bought
5	Discount	float	Discount of the product bought

2.2.17 Purchase Payment History table

No	Field Name	Type	Description
1	PaymentId	int	ID of the purchase order payment Primary Key Auto increment
2	Payment Date	datetime	Date of the purchase order payment
3	Payment Amount	decimal(15, 0)	The amount of money paid
4	PurchaseId	int	ID of the purchase order being paid Foreign key reference to "PurchaseOrderId" of "PurchaseOrder" table
5	StaffId	int	ID of the staff handling the payment Foreign key reference to "StaffId" of "Staff" table

2.2.18 Supplier table

No	Field Name	Type	Description
1	SupplierId	int	ID of the supplier Primary Key Auto increment
2	Address	nvarchar(150)	Supplier's detailed address
3	Supplier Name	nvarchar(300)	Supplier's name
4	Phone	varchar(15)	Supplier's phone number
5	Status	bit	Supplier's status True for Active False for Inactive
6	Description	nvarchar(1000)	Supplier's description
7	Email	nvarchar(100)	Supplier's email address
8	Province Code	int	The province code of supplier's address (used by API)
9	Province	nvarchar(50)	The province name of supplier's address
10	DistrictCode	int	The district code of supplier's address (used by API)

No	Field Name	Type	Description
11	District	nvarchar(50)	The district name of supplier's address
12	WardCode	int	The ward code of supplier's address (used by API)
13	Ward	nvarchar(50)	The ward name of supplier's address

2.2.19 Shipping Info table

No	Field Name	Type	Description
1	ShipInfoId	int	ID of the shipping information Primary Key Auto increment
2	Phone	nchar(10)	Phone number in shipping information
3	ShipAddress	nvarchar(200)	Address in shipping information
4	CustomerId	int	The Id of customer this shipping information belongs to Foreign key reference to "CustomerId" of "Customer" table
5	Status	bit	Shipping's information status True for Active False for Inactive
6	Customer Name	nvarchar(50)	Recipient name in shipping information
7	Province Code	int	The province code of shipping information's address (used by API)
8	Province	nvarchar(50)	The province name of shipping information's address
9	DistrictCode	int	The district code of shipping information's address (used by API)
10	District	nvarchar(50)	The district name of shipping information's address
11	WardCode	int	The ward code of shipping information's address (used by API)
12	Ward	nvarchar(50)	The ward name of shipping information's address

2.2.20 Cart Detail table

No	Field Name	Type	Description
1	CustomerId	int	Id of the customer this cart detail belongs to Primary Key Foreign key reference to "CustomerId" of "Customer" table
2	ProductId	int	Id of the product in the cart Primary Key Foreign key reference to "ProductId" of "Product" table
3	Quantity	int	The number of said product in cart

2.2.21 Product table

No	Field Name	Type	Description
1	ProductId	int	Id of product Primary Key Auto increment
2	Name	nvarchar(100)	Name of product
3	Barcode	nvarchar(50)	Barcode of product Unique
4	Unit	nvarchar(50)	The unit this product is calculated in (individual item, box, package,...)
5	UnitInStock	int	The number of product units currently in storage
6	Purchase Price	decimal(7, 0)	Product's purchase price
7	RetailPrice	int	Product's retail price
8	Retail Discount	float	Product's discount when sold to retail customers
9	Wholesale Price	int	Product's wholesale price
10	Wholesale Discount	float	Product's discount when sold to wholesale customers
11	Size	nvarchar(50)	Product's size
12	Weight	nvarchar(50)	Product's weight
13	Description	ntext	Product's description
14	Status	bit	Product's status True for Active False for Inactive
15	IsBook	bit	Product's subtype - Book or Stationery True for Book False for Stationery
16	SubCategory Id	int	Id of Product's subcategory Foreign key reference to "SubCategoryId" of "SubCategory" table
17	Available Unit	int	The number of product units that can be sold

2.2.22 Book table

No	Field Name	Type	Description
1	ProductId	int	Id of this book product Primary Key Foreign key reference to "ProductId" of "Product" table
2	Pages	int	Number of pages this book has
3	Language	nvarchar(50)	Language of this book
4	Publisher	nvarchar(100)	This book's publisher's name

2.2.23 Author table

No	Field Name	Type	Description
1	AuthorId	int	Id of author Primary Key Auto increment
2	AuthorName	nvarchar(50)	Name of author
3	Description	ntext	Author's description
4	Status	bit	Author's status True for Active False for Inactive

2.2.24 Book-Author table

No	Field Name	Type	Description
1	BookId	int	Id of book Primary Key Foreign key reference to "ProductId" of "Book" table
2	AuthorId	int	Id of author Primary Key Foreign key reference to "AuthorId" of "Author" table

2.2.25 Stationery table

No	Field Name	Type	Description
1	ProductId	int	Id of this stationery product Primary Key Foreign key reference to "ProductId" of "Product" table
2	Material	nvarchar(50)	Stationery's material
3	Origin	nvarchar(50)	Stationery's origin country
4	Brand	nvarchar(100)	Stationery's brand

2.2.26 Image table

No	Field Name	Type	Description
1	ImageId	int	Id of image Primary Key Auto increment
2	ImageName	nvarchar(50)	Name of the image file
3	ProductId	int	ID of the product this image belongs to Foreign key reference to "ProductId" of "Product" table
4	Status	bit	Image's status True for Active False for Inactive

2.2.27 Category table

No	Field Name	Type	Description
1	CategoryId	int	ID of category Primary Key Auto increment
2	Category Name	nvarchar(50)	Name of category

2.2.28 Sub-category table

No	Field Name	Type	Description
1	SubCategory Id	int	ID of subcategory Primary Key Auto increment
2	SubCategory Name	nvarchar(50)	Name of subcategory
3	CategoryId	int	ID of the category this subcategory belongs to Foreign key reference to "CategoryId" of "Category" table

2.2.29 Exchange table

No	Field Name	Type	Description
1	ExchangeId	int	ID of exchange Primary Key Auto increment
2	ProductId	int	ID of the product converted into Foreign key reference to "ProductId" of "Product" table
3	BaseProduct Id	int	ID of the product being converted Foreign key reference to "ProductId" of "Product" table
4	Quantity BaseProduct	int	The number of base product units 1 new product unit consists of

2.2.30 Review table

No	Field Name	Type	Description
1	ReviewId	int	ID of review Primary Key Auto increment
2	Rating	float	Rating on the scale of 5 in the review
3	Comment	text	The content of review's comment
4	AccountId	int	ID of the customer making the review
5	ProductId	int	ID of the product this review is for Foreign key reference to "ProductId" of "Product" table
6	Status	bit	Review's status True for Active False for Inactive
7	CreateDate	datetime	The time and date this review is added

3. Detailed Design

3.1 Authentication and Authorization functions

3.1.1 Class Diagram

InputModel
+ Password: string
+ Email: string
+ RememberMe: bool

LoginModel
SignInManager<IdentityUser> _signInManager;
UserManager<IdentityUser> _userManager
ILogger<LoginModel> _logger
InputModel Input
string ErrorMessage
+ ProductController(productRespository: IProductRespository, authorRepository: IAuthorRepsitory, categoryRepository: ICategoryRepository, imageRepository: IImageRepository)
+ <<async>> OnGetAsync(string returnUrl = null): Task
+ <<async>> OnPostAsync(string returnUrl = null): Task<IActionResult>

InputModel
+ Email: string
+ Password: string
+ ConfirmPassword: string
+ Role: string?
RoleList: IEnumerable<SelectListItem>

RegisterModel
SignInManager<IdentityUser> _signInManager;
UserManager<IdentityUser> _userManager
RoleManager<IdentityRole> _roleManager
IUserStore<IdentityUser> _userStore
IUserEmailStore<IdentityUser> _emailStore
+ public RegisterModel(UserManager<IdentityUser> userManager, RoleManager<IdentityRole> roleManager, IUserStore<IdentityUser> userStore, SignInManager<IdentityUser> signInManager, ILogger<RegisterModel> logger, IEmailSender emailSender, IWebHostEnvironment webHostEnvironment)
+ <<async>> OnGetAsync(string returnUrl = null): Task
+ <<async>> OnPostAsync(string returnUrl = null): Task<IActionResult>

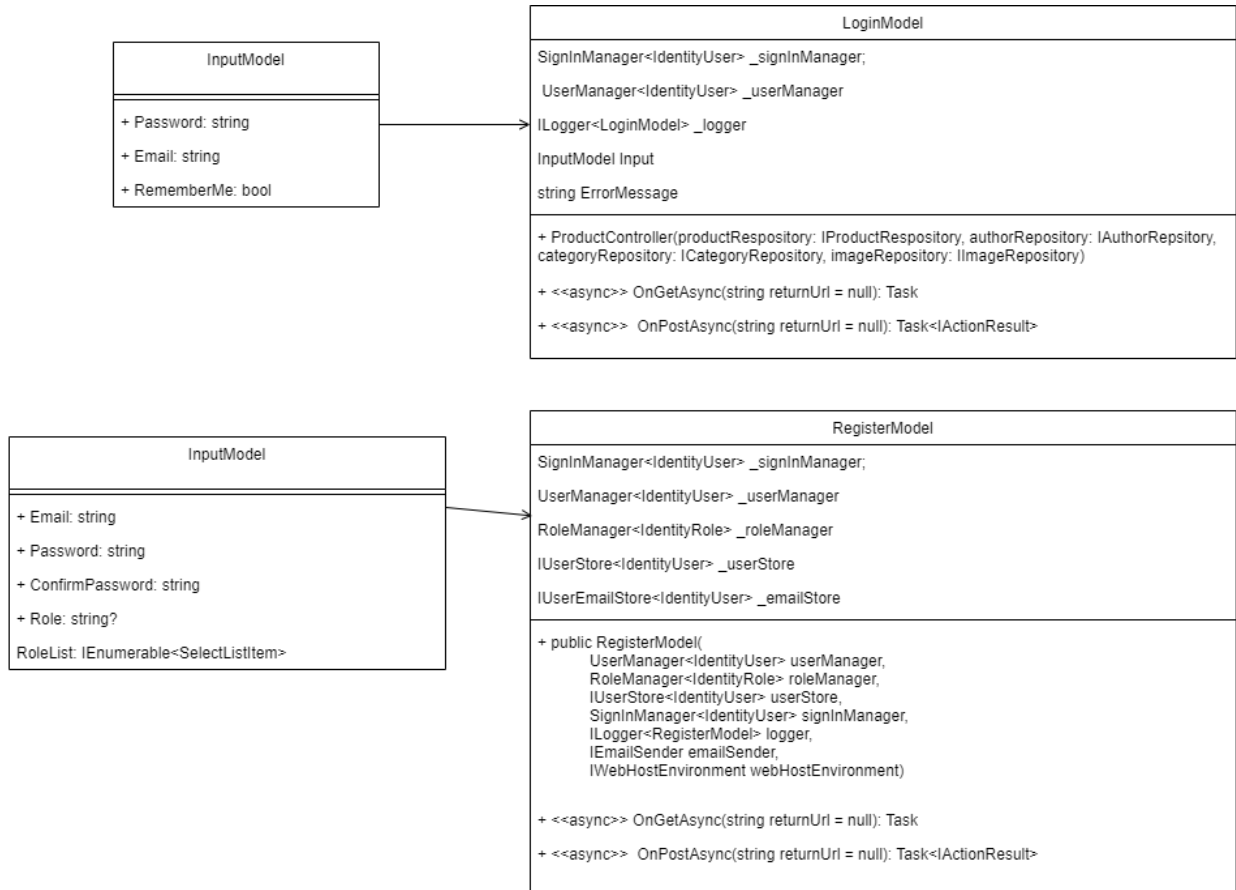*Figure 3.1.1.1: Authentication and Authorization functions class diagram*
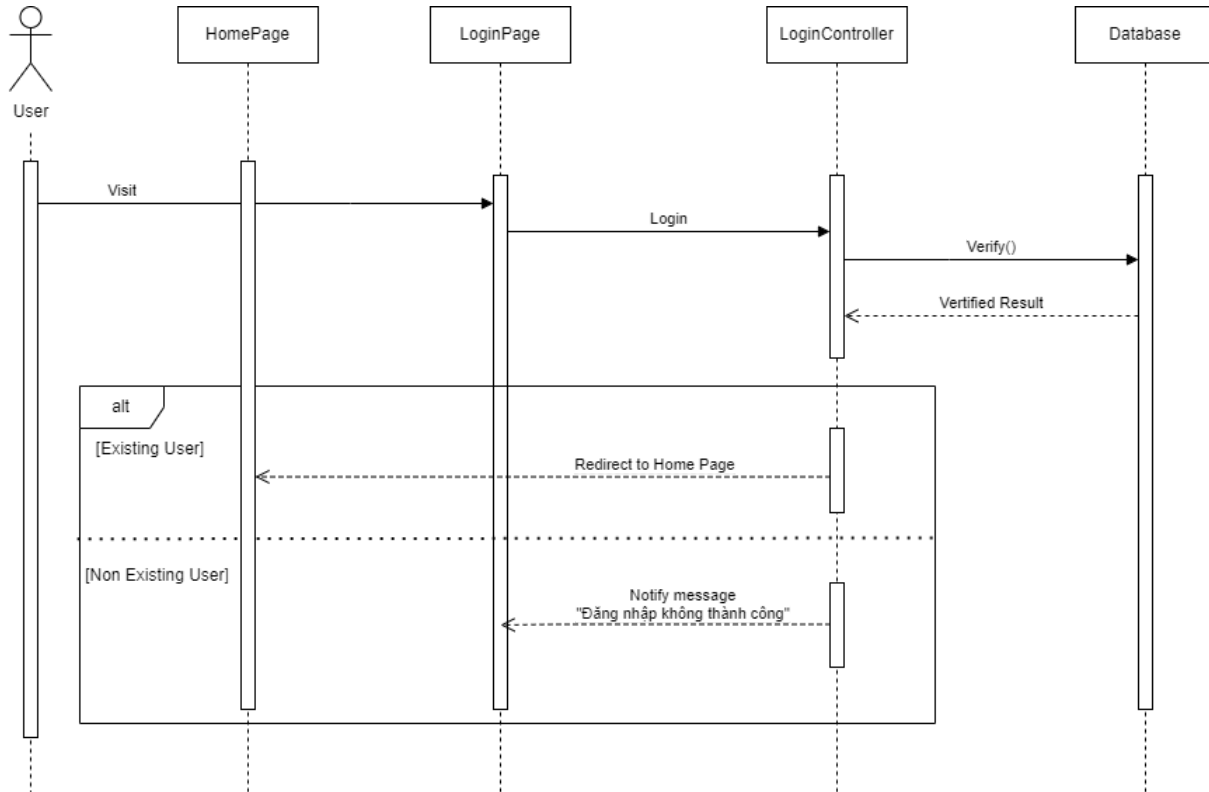
3.1.2 Login

Figure 3.1.2.1: Login sequence diagram
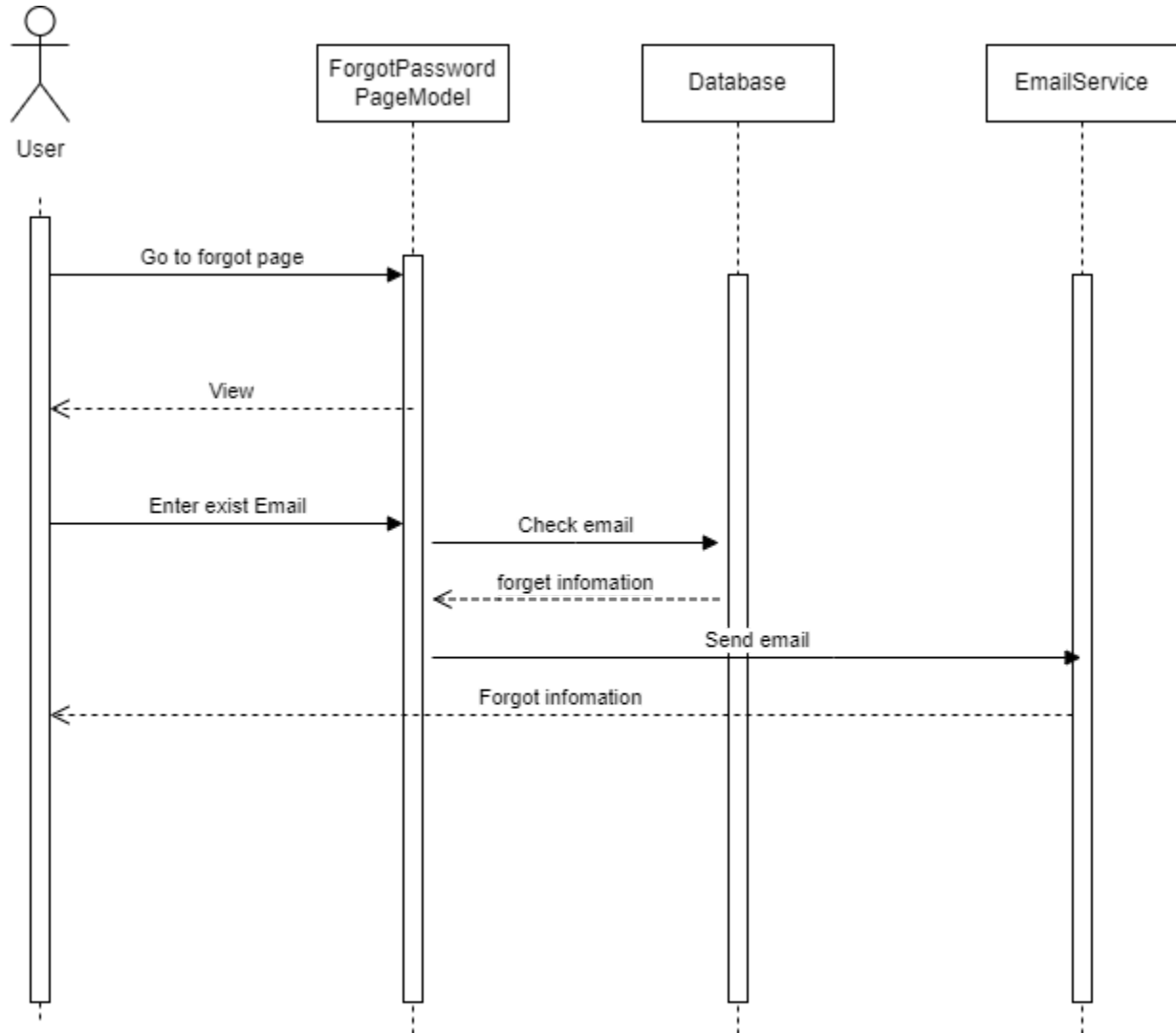
3.1.3. Forgot Password

Figure 3.1.3.1: Forgot password sequence diagram
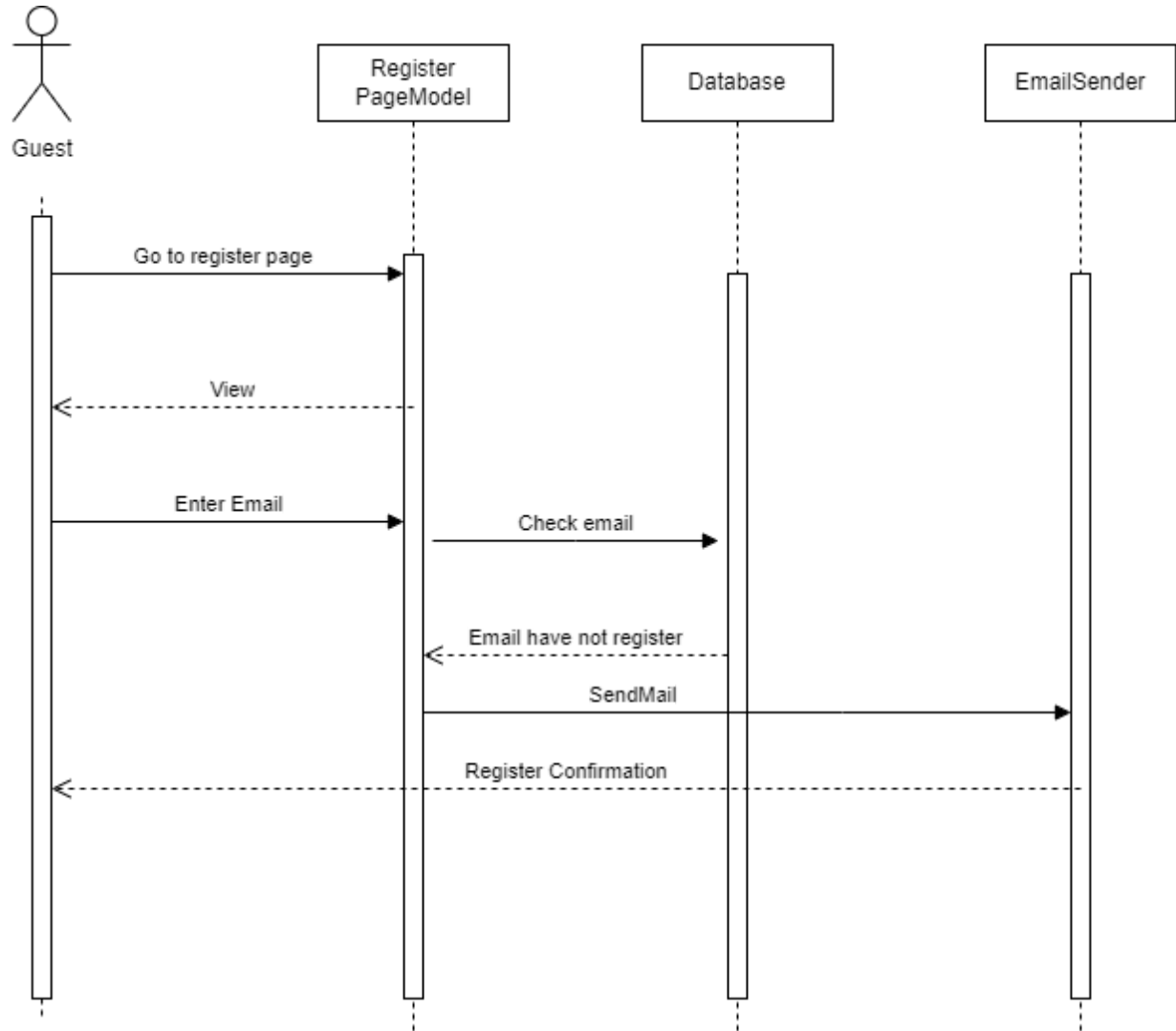
3.1.4. Register

Figure 3.1.4.1: Register sequence diagram

3.1.5 Email Re-confirm
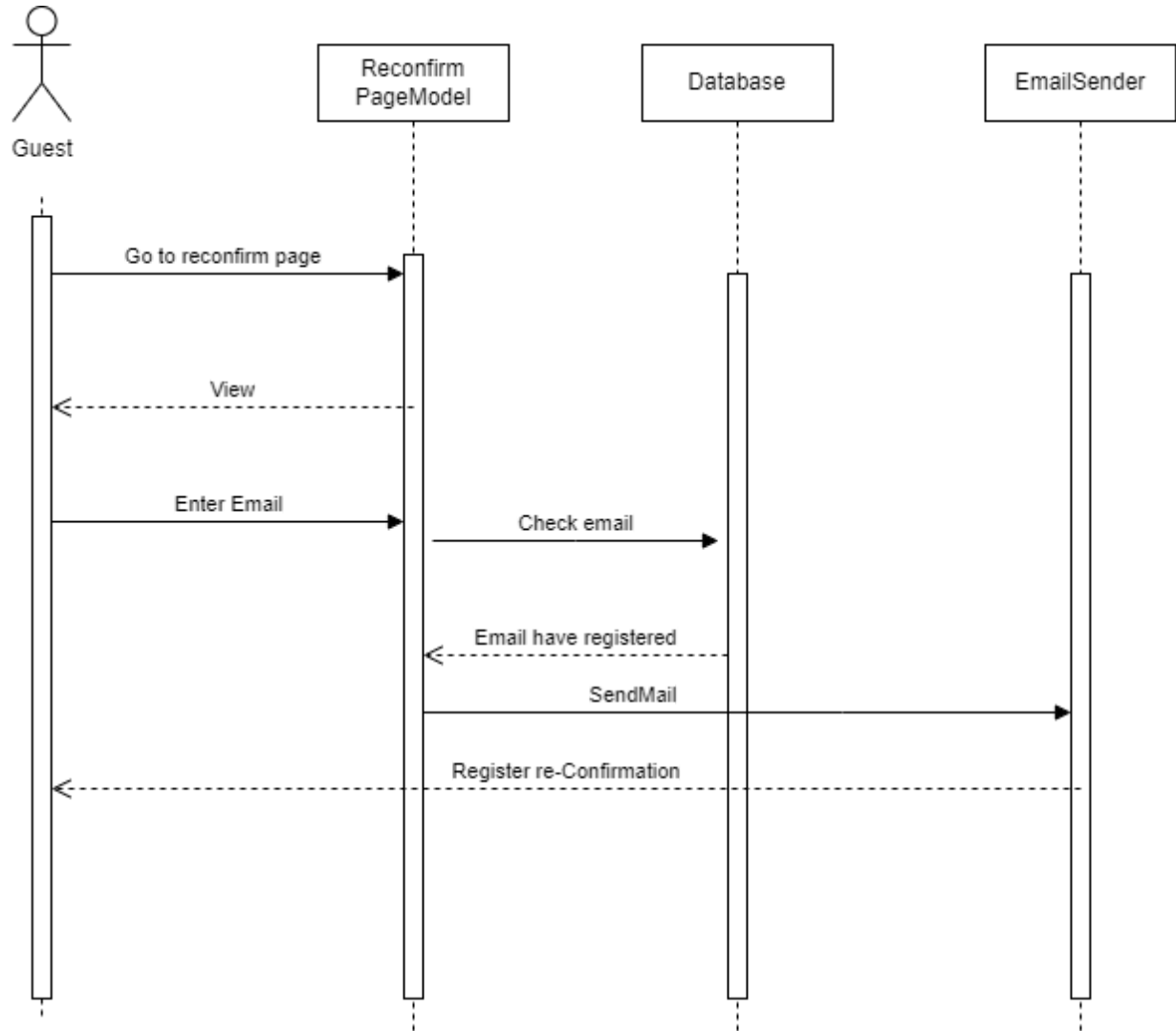
Figure 3.1.5.1: E-mail Re-confirm sequence diagram

3.1.6 Logout
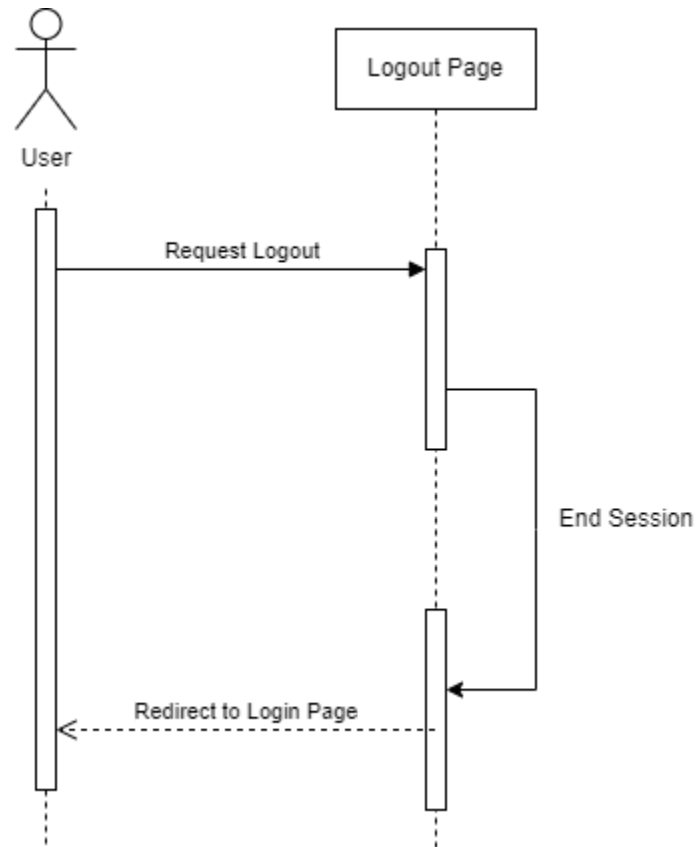
Figure 3.1.6.1: Logout sequence diagram

3.2 Category and Subcategory Management

3.2.1 Class Diagram

<<interface>>
ICategoryResposirory

+ GetAllCategoriesAsync(): Task<List<Category>>

+ GetAllCategoriesAllInfoAsync():Task<List<Category>>

+ AddCategory(Category c): Task

+ UpdateCategory(Category c): Task

+ DeleteCategory(int cid): Task

+ GetCategoryById(int cid): Task<Category>

CategotyResposirory

CategoriesController

- categoryRespository: ICategoryRespository

+ CategoriesController(ICategoryRespository categoryRespository)

+ <<async>> Index() : Task<IActionResult>

+ <<async>> Create(Category category): Task<IActionResult>

+ <<async>> Edit(int? id)): Task<IActionResult>

+ <<async>>Edit(int id, Category category): Task<IActionResult>

+ <<async>> Delete(int? id): Task<IActionResult>

+ <<async>> DeleteConfirmed(int id): Task<IActionResult>

<<interface>>
ISubCategoryResposirory

+ AddSubCategory(SubCategory sub): Task

+ UpdateSubCategory(SubCategory sub): Task

+ DeleteSubCategory(int subId): Task

SubCategoryResposirory

SubCategoriesController

- subCategoryRespository: ISubCategoryRespository

SubCategoriesController(ICategoryRespository categoryRespository)

+ <<async>> Index() : Task<IActionResult>

+ <<async>> Create(SubCategory subcategory): Task<IActionResult>

+ <<async>> Edit(int? id)): Task<IActionResult>

+ <<async>>Edit(int id, SubCategory subcategory): Task<IActionResult>

+ <<async>> Delete(int? id): Task<IActionResult>

+ <<async>> DeleteConfirmed(int id): Task<IActionResult>

Figure 3.2.1.1: Category and Subcategory Management class diagram

3.2.2 View list category and subcategories
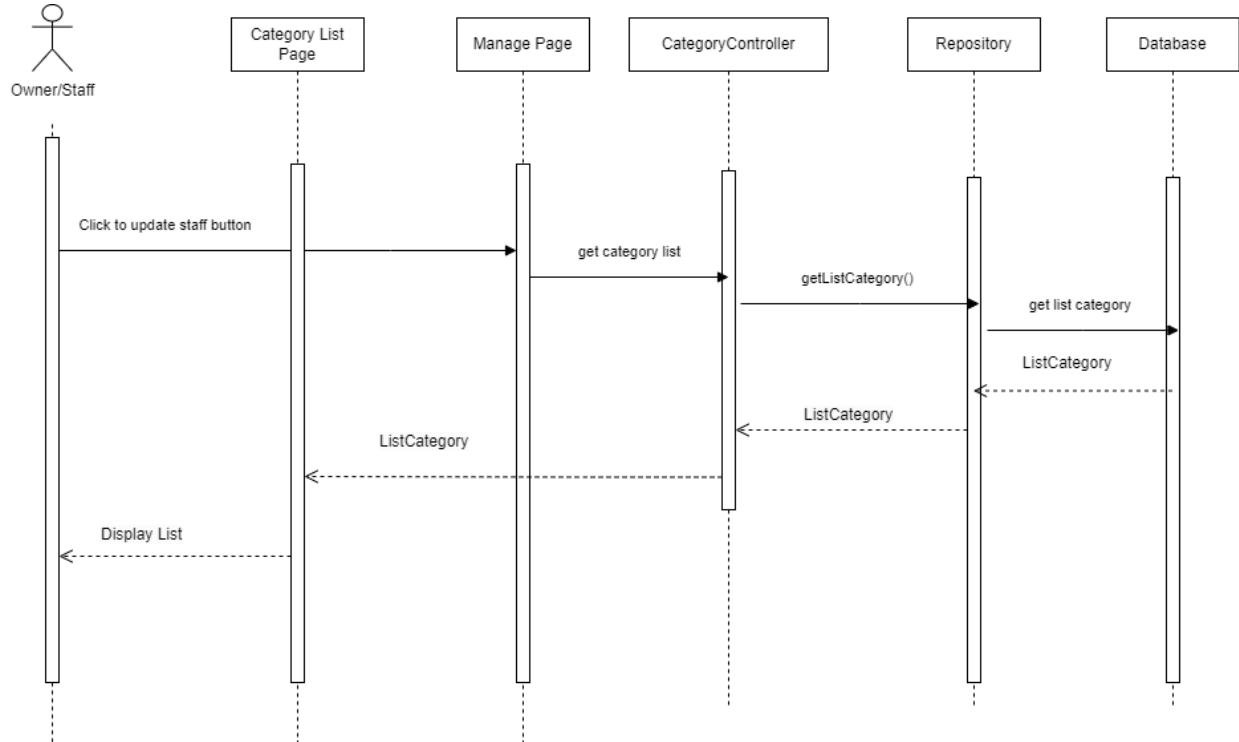
Figure 3.2.2.1: View list category and subcategories sequence diagram

3.2.2 Add category
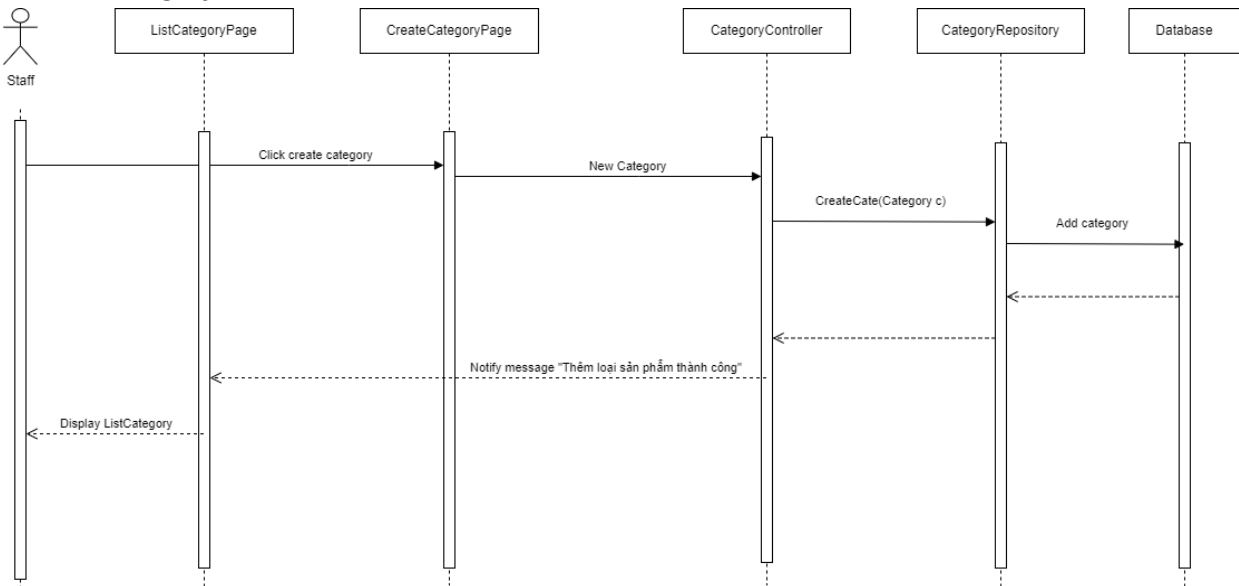
Figure 3.2.2.1: Add category sequence diagram

3.2.3 Update category

Figure 3.2.3.1: Update category sequence diagram

3.2.4 Delete category

Figure 3.2.4.1: Delete category sequence diagram

3.2.5 Add subcategory

Figure 3.2.5.1: Add subcategory sequence diagram

3.2.6 Update subcategory

Figure 3.2.6.1: Update subcategory sequence diagram

3.2.7 Delete subcategory

Figure 3.2.7.1: Delete subcategory sequence diagram
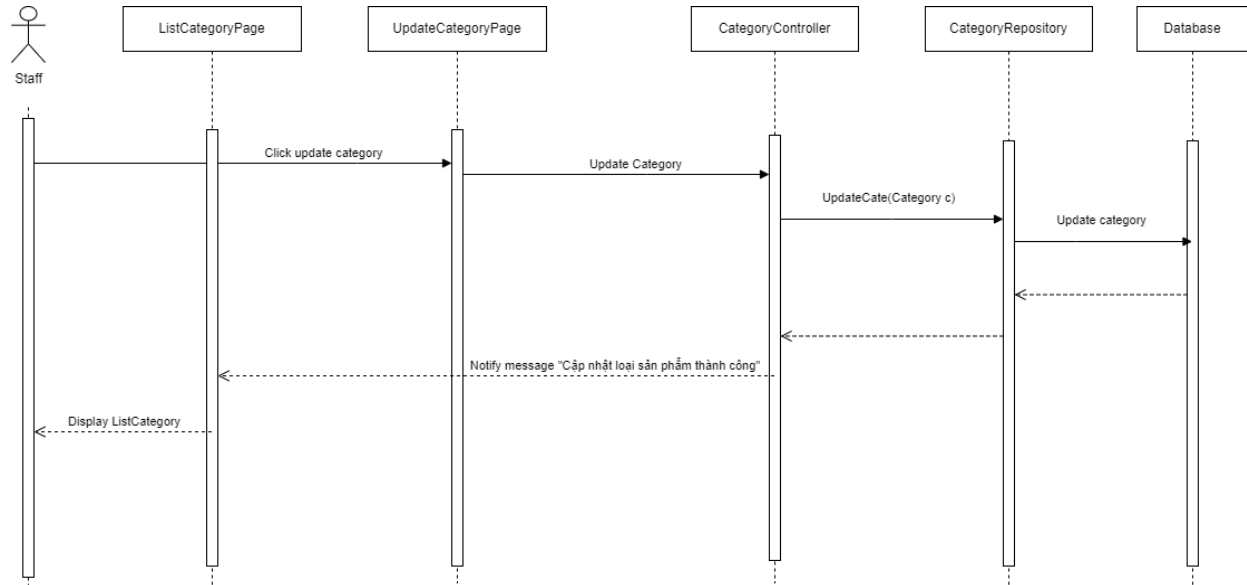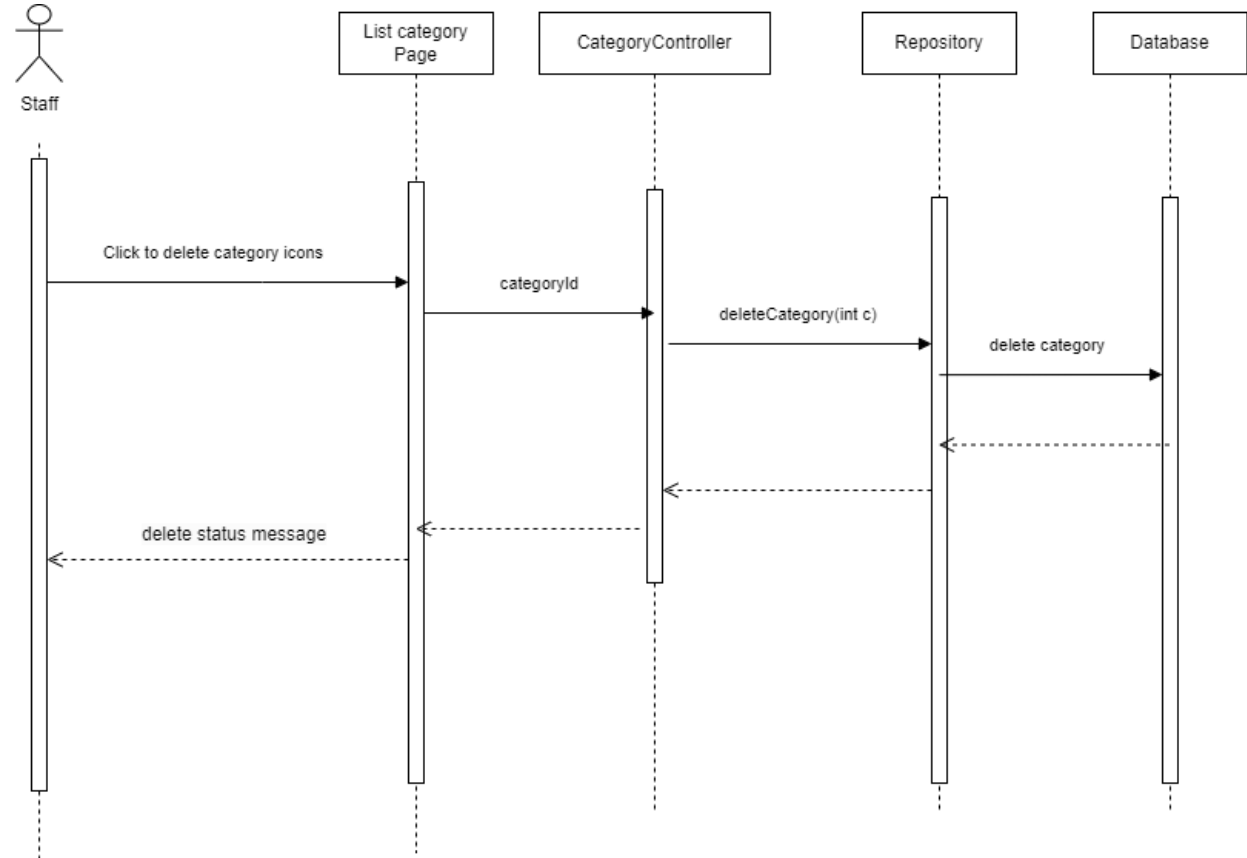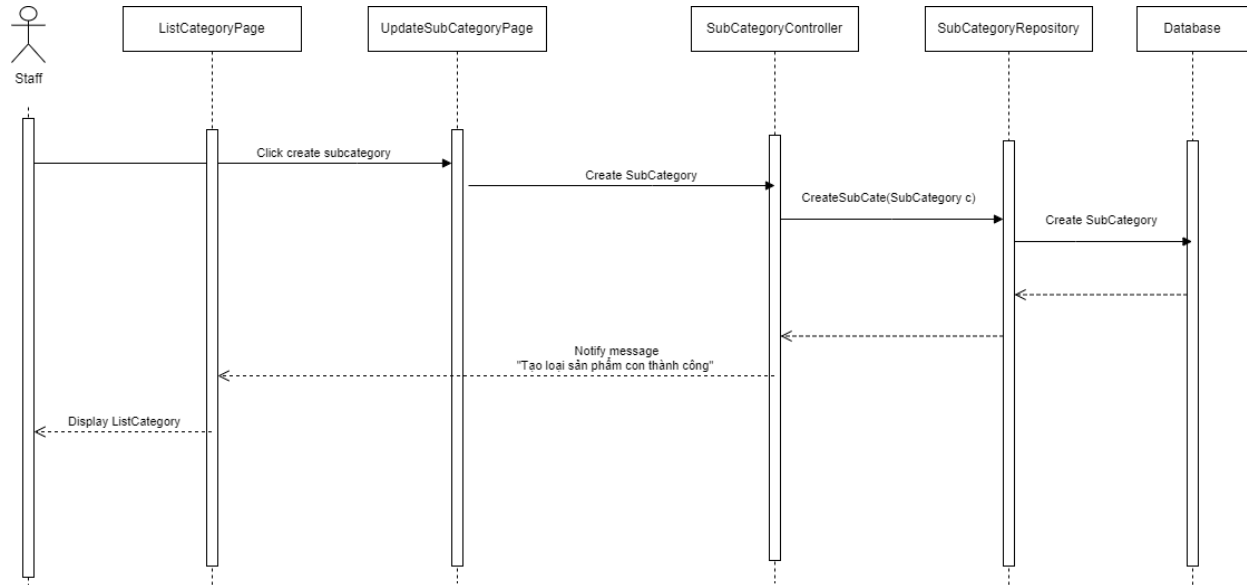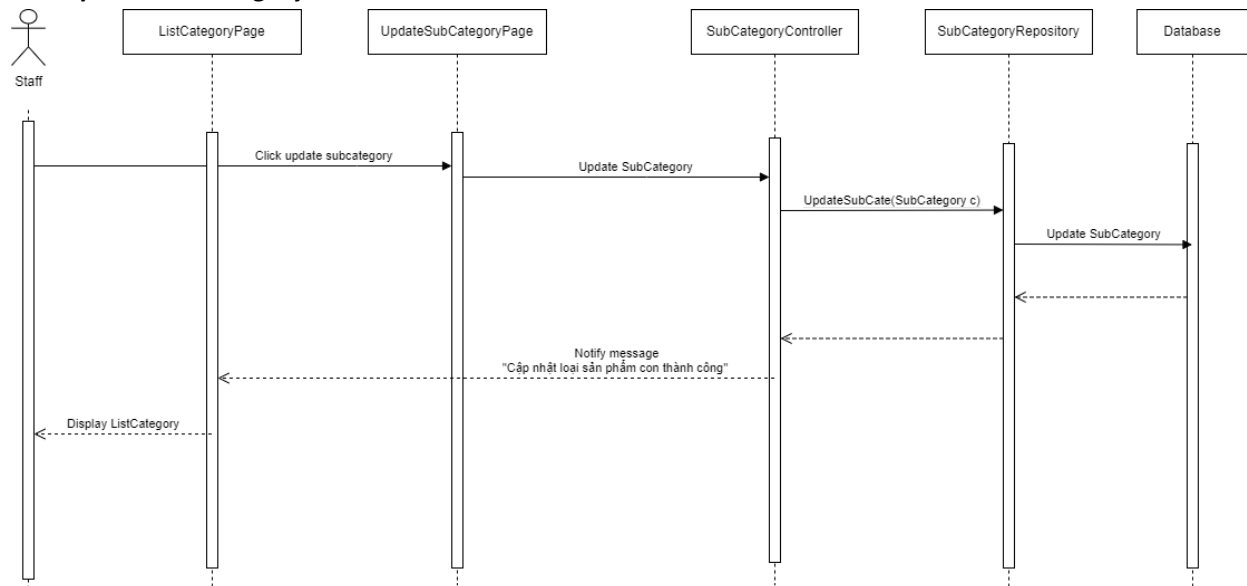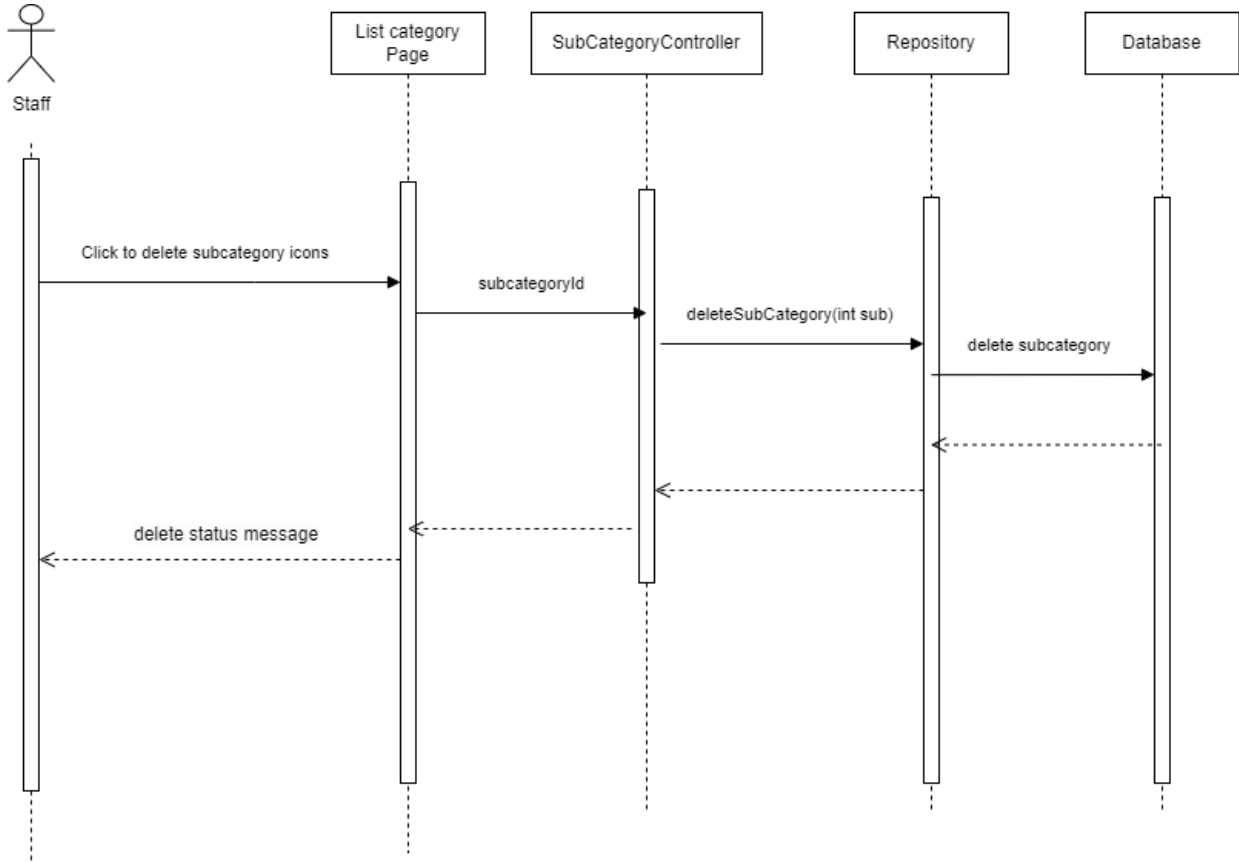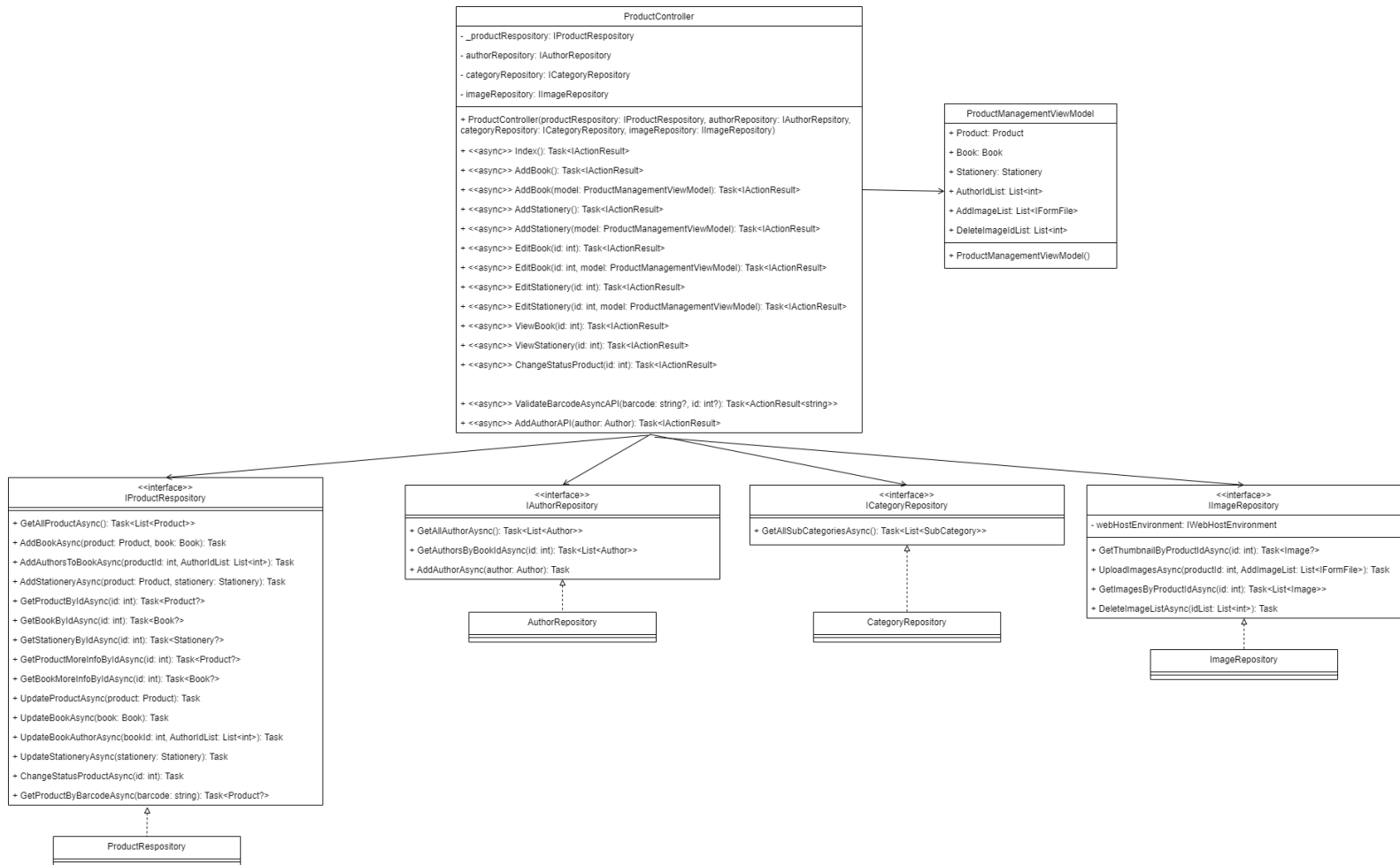
3.3 Product Management

3.3.1 Class Diagram

ProductController

- _productRespository: IProductRespository
- authorRepository: IAuthorRepository
- categoryRepository: ICategoryRepository
- imageRepository: IImageRepository

+ ProductController(productRespository: IProductRespository, authorRepository: IAuthorRepsitory, categoryRepository: ICategoryRepository, imageRepository: IImageRepository)
+ <<async>> Index(): Task<IActionResult>
+ <<async>> AddBook(): Task<IActionResult>
+ <<async>> AddBook(model: ProductManagementViewModel): Task<IActionResult>
+ <<async>> AddStationery(): Task<IActionResult>
+ <<async>> AddStationery(model: ProductManagementViewModel): Task<IActionResult>
+ <<async>> EditBook(id: int): Task<IActionResult>
+ <<async>> EditBook(id: int, model: ProductManagementViewModel): Task<IActionResult>
+ <<async>> EditStationery(id: int): Task<IActionResult>
+ <<async>> EditStationery(id: int, model: ProductManagementViewModel): Task<IActionResult>
+ <<async>> ViewBook(id: int): Task<IActionResult>
+ <<async>> ViewStationery(id: int): Task<IActionResult>
+ <<async>> ChangeStatusProduct(id: int): Task<IActionResult>

+ <<async>> ValidateBarcodeAsyncAPI(barcode: string?, id: int?): Task<ActionResult<string>>
+ <<async>> AddAuthorAPI(author: Author): Task<IActionResult>

ProductManagementViewModel

+ Product: Product
+ Book: Book
+ Stationery: Stationery
+ AuthorIdList: List<int>
+ AddImageList: List<IFormFile>
+ DeleteImageIdList: List<int>

+ ProductManagementViewModel()

<<interface>> IProductRespository

+ GetAllProductAsync(): Task<List<Product>>
+ AddBookAsync(product: Product, book: Book): Task
+ AddAuthorsToBookAsync(productId: int, AuthorIdList: List<int>): Task
+ AddStationeryAsync(product: Product, stationery: Stationery): Task
+ GetProductByIdAsync(id: int): Task<Product?>
+ GetBookByIdAsync(id: int): Task<Book?>
+ GetStationeryByIdAsync(id: int): Task<Stationery?>
+ GetProductMoreInfoByIdAsync(id: int): Task<Product?>
+ GetBookMoreInfoByIdAsync(id: int): Task<Book?>
+ UpdateProductAsync(product: Product): Task
+ UpdateBookAsync(book: Book): Task
+ UpdateBookAuthorAsync(bookId: int, AuthorIdList: List<int>): Task
+ UpdateStationeryAsync(stationery: Stationery): Task
+ ChangeStatusProductAsync(id: int): Task
+ GetProductByBarcodeAsync(barcode: string): Task<Product?>

ProductRespository

<<interface>> IAuthorRepository

+ GetAllAuthorAsync(): Task<List<Author>>
+ GetAuthorsByBookIdAsync(id: int): Task<List<Author>>
+ AddAuthorAsync(author: Author): Task

AuthorRepository

<<interface>> ICategoryRepository

+ GetAllSubCategoriesAsync(): Task<List<SubCategory>>

CategoryRepository

<<interface>> IImageRepository

- webHostEnvironment: IWebHostEnvironment
+ GetThumbnailByProductIdAsync(id: int): Task<Image?>
+ UploadImagesAsync(productId: int, AddImageList: List<IFormFile>): Task
+ GetImagesByProductIdAsync(id: int): Task<List<Image>>
+ DeleteImageListAsync(idList: List<int>): Task

ImageRepository

Figure 3.3.1.1: Product Management functions class diagram
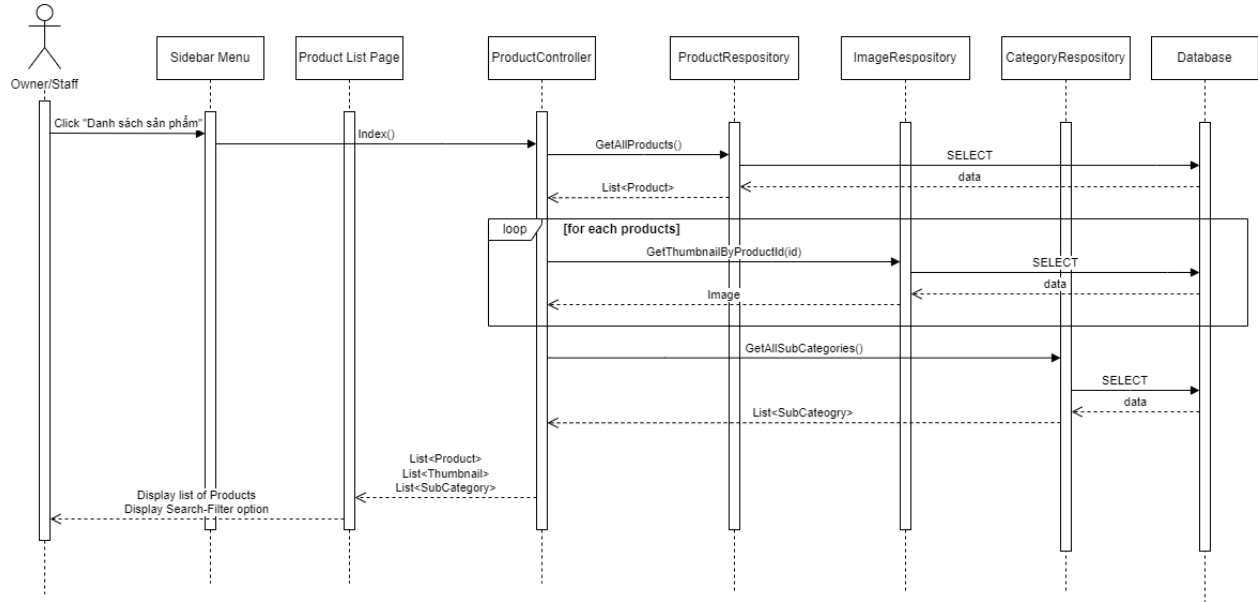
3.3.2 View List Products

Figure 3.3.2.1: View List Products sequence diagram
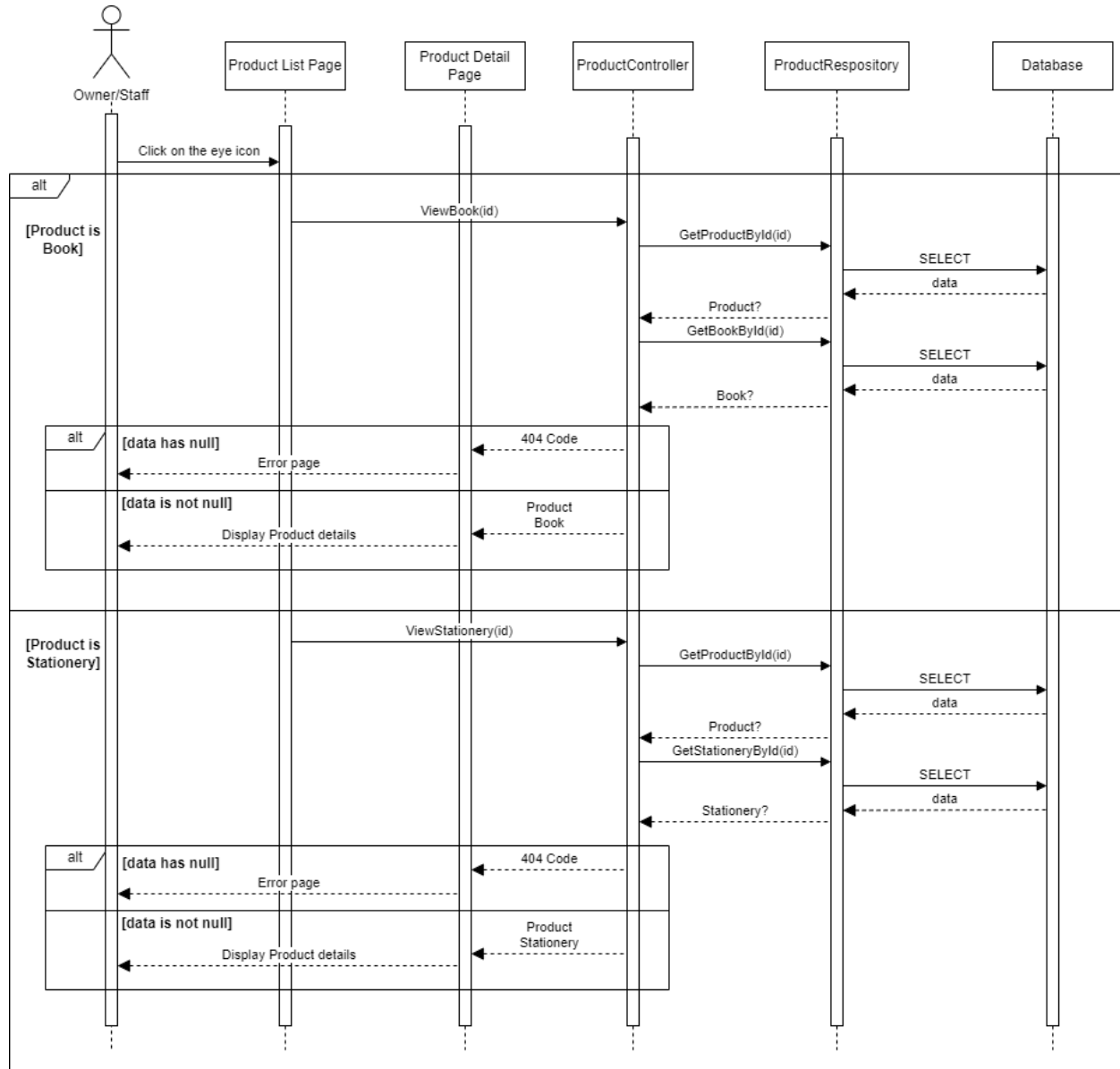
3.3.3 View Product Details

Figure 3.3.3.1: View Product Details sequence diagram

3.3.4 Create Product

Figure 3.3.4.1: Create Product sequence diagram

3.3.5 Update Product
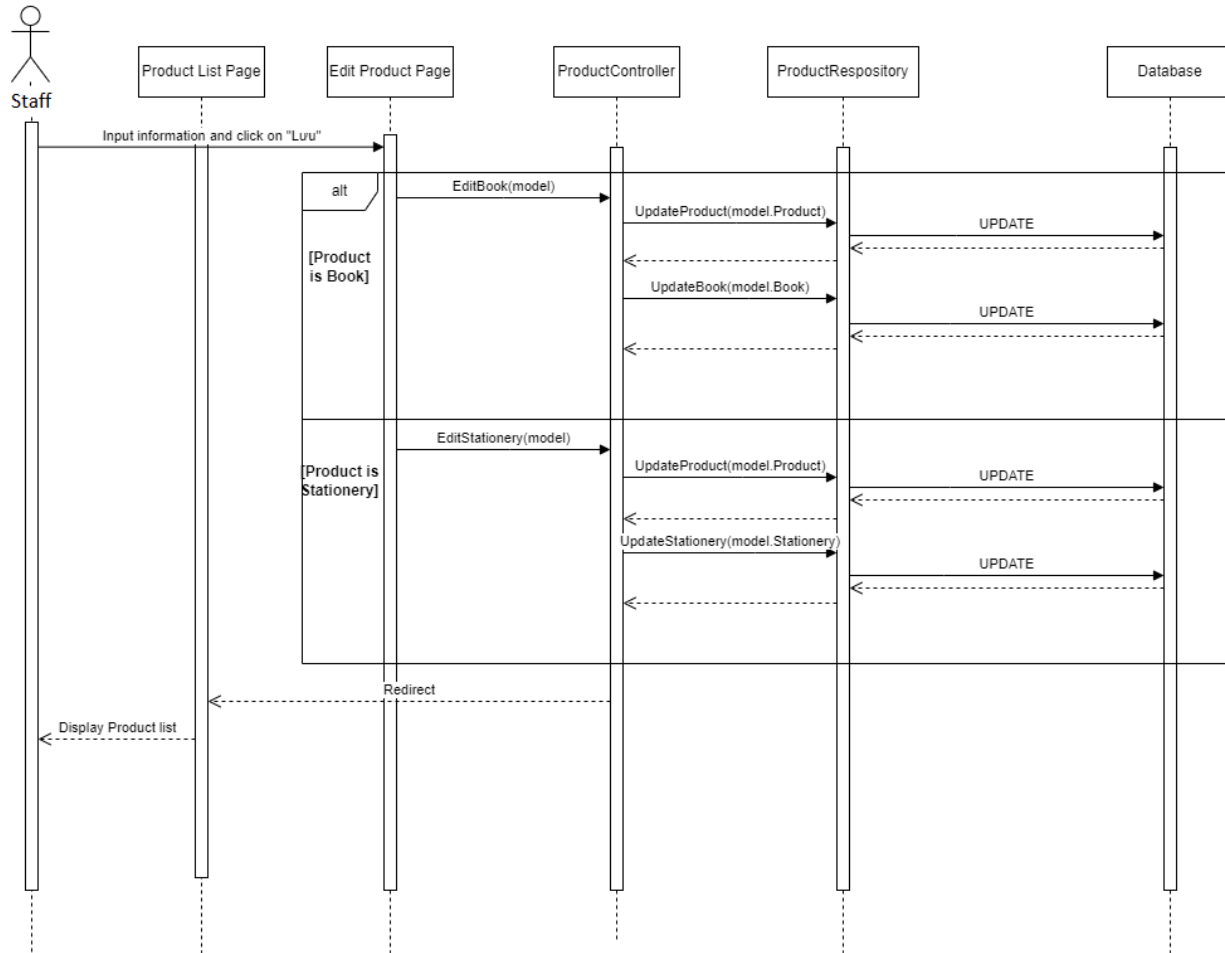
Figure 3.3.5.1: Update Product sequence diagram

3.4 Author Management

3.4.1 Class Diagram

Figure 3.4.1.1: Author Management functions class diagram

3.4.2 View list of Authors
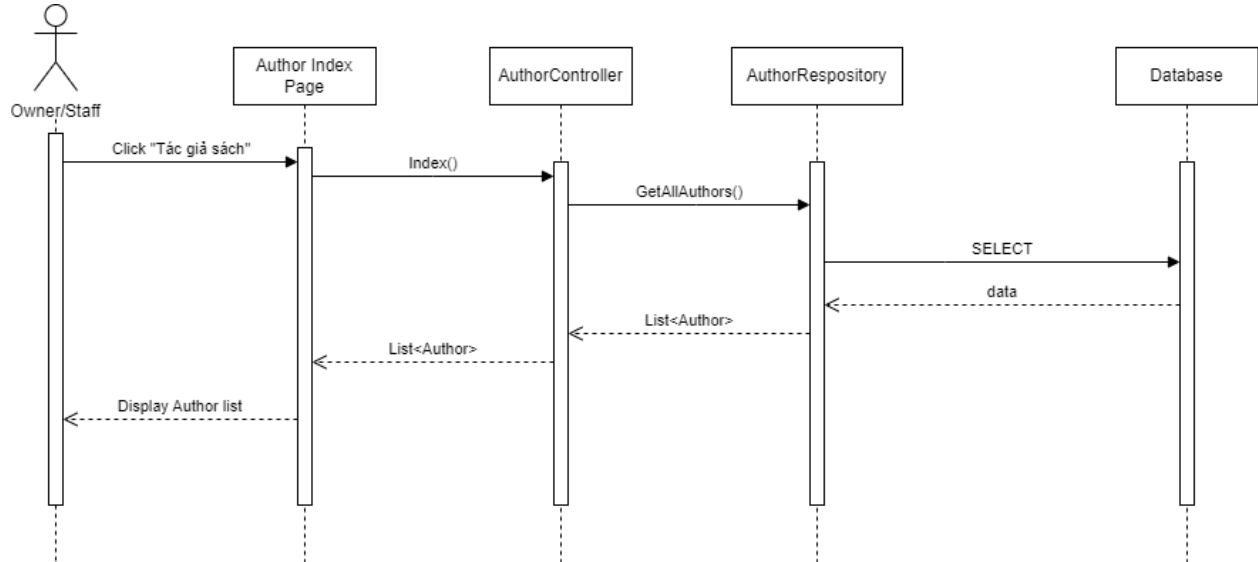
Figure 3.4.2.1: View list of Authors sequence diagram

3.4.3 Add new Author
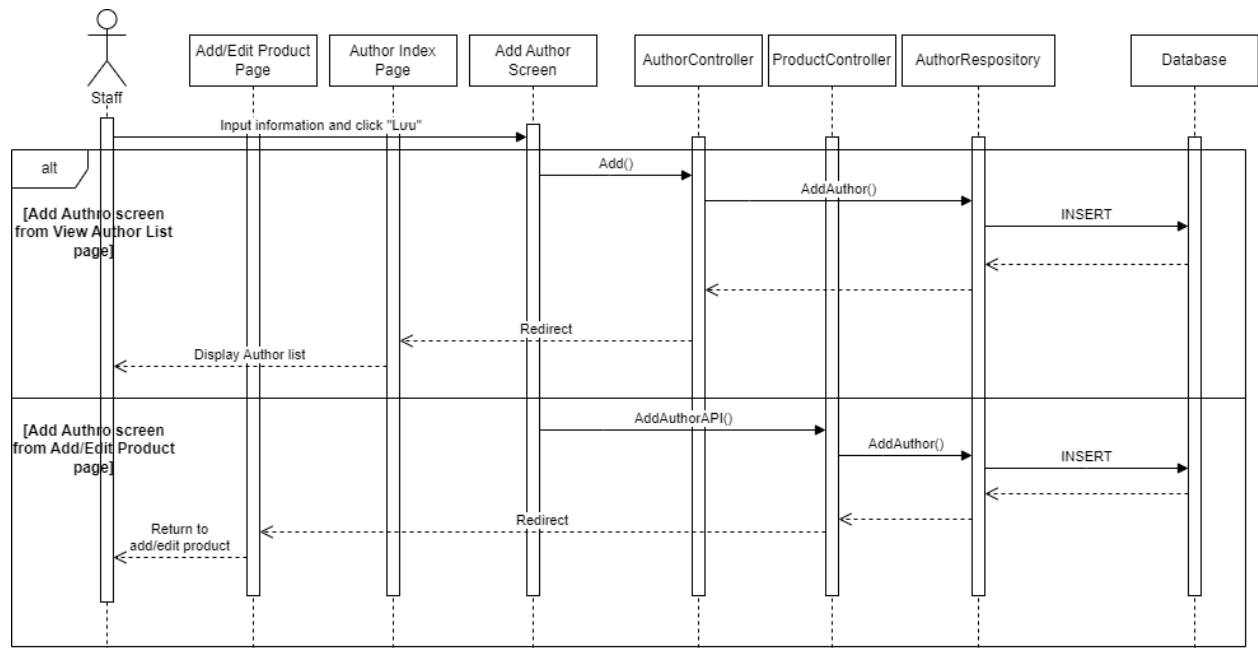
Figure 3.4.3.1: Add new Author sequence diagram

3.4.4 Edit Author
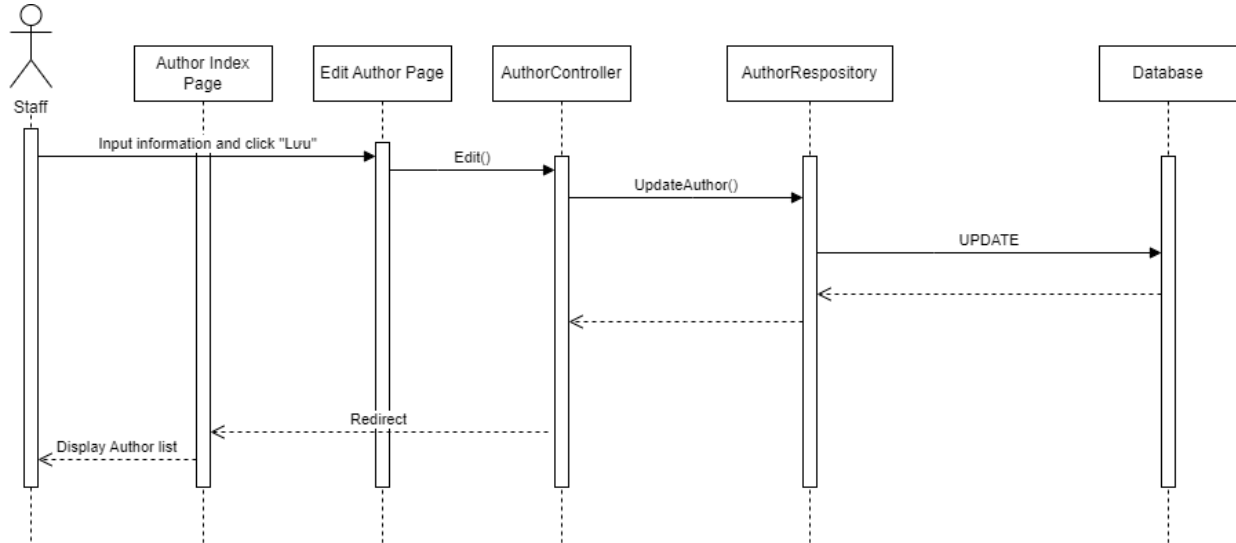
Figure 3.4.4.1: Edit Author sequence diagram

3.4.5 Change Status Author
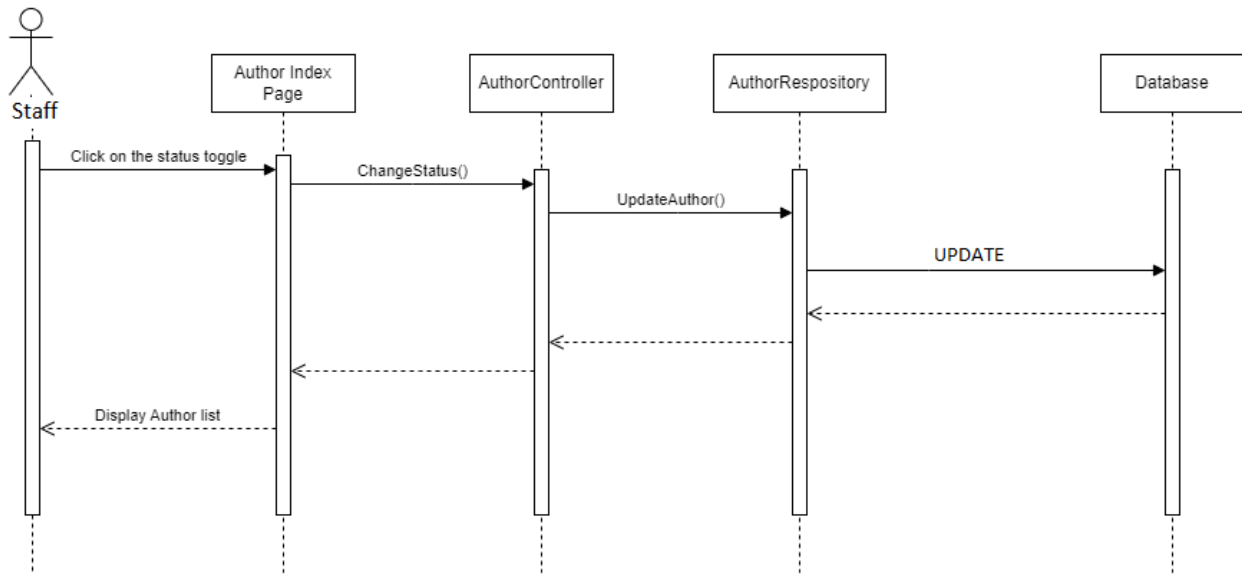
Figure 3.4.5.1: Change Status Author sequence diagram

3.5 Buy Products

3.5.1 Class Diagram
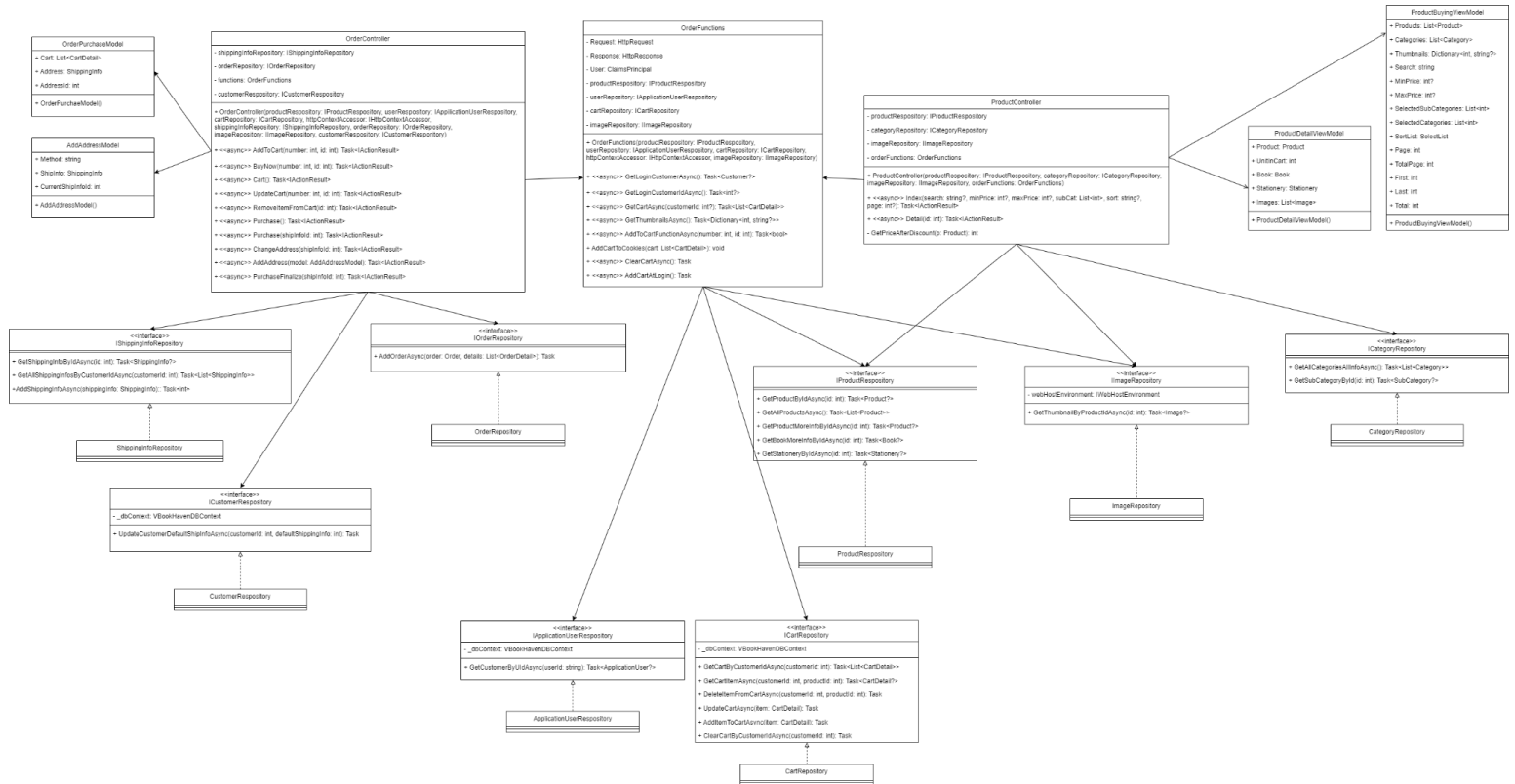
Figure 3.5.1.1: Buy Products functions class diagram

3.5.2 View List Products (Customer)
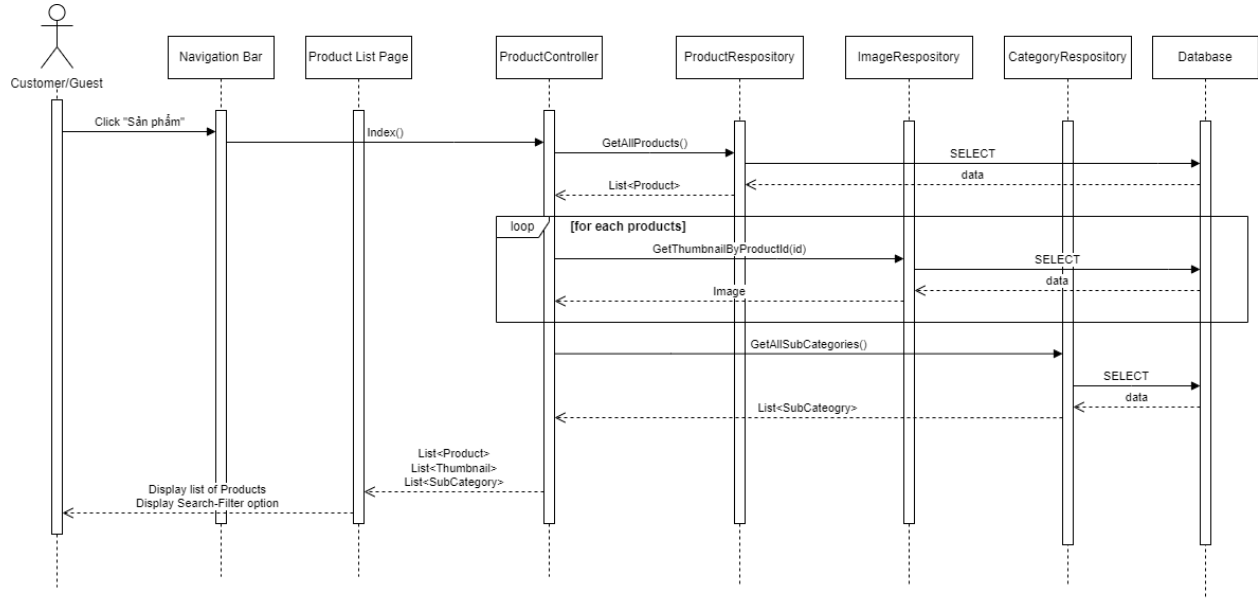
Figure 3.5.2.1: View List Products (Customer) functions sequence diagram

3.5.3 View Product Details (Customer)
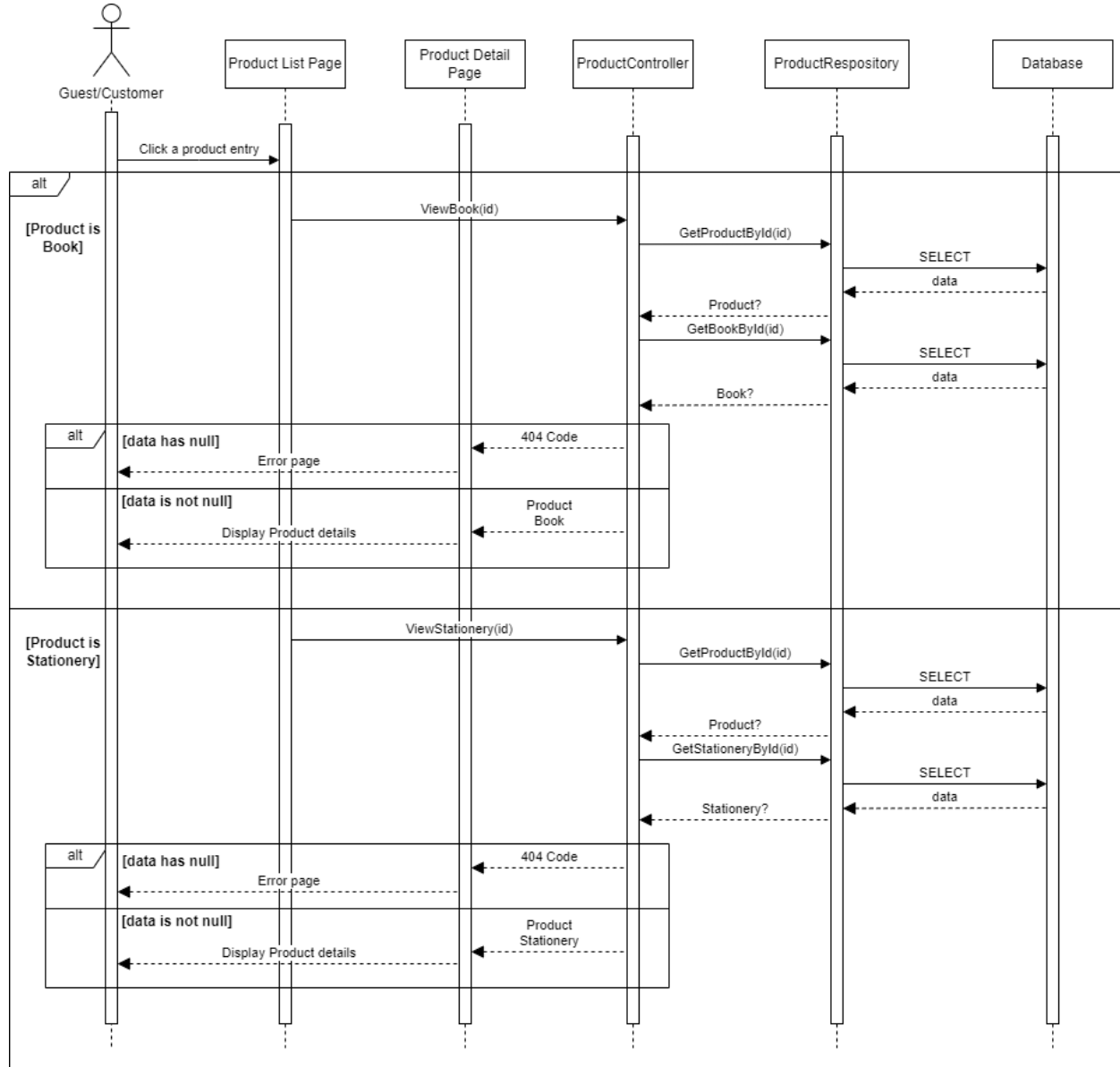
Figure 3.5.3.1: View Product Details (Customer) functions sequence diagram

3.5.4 Add Product to Cart
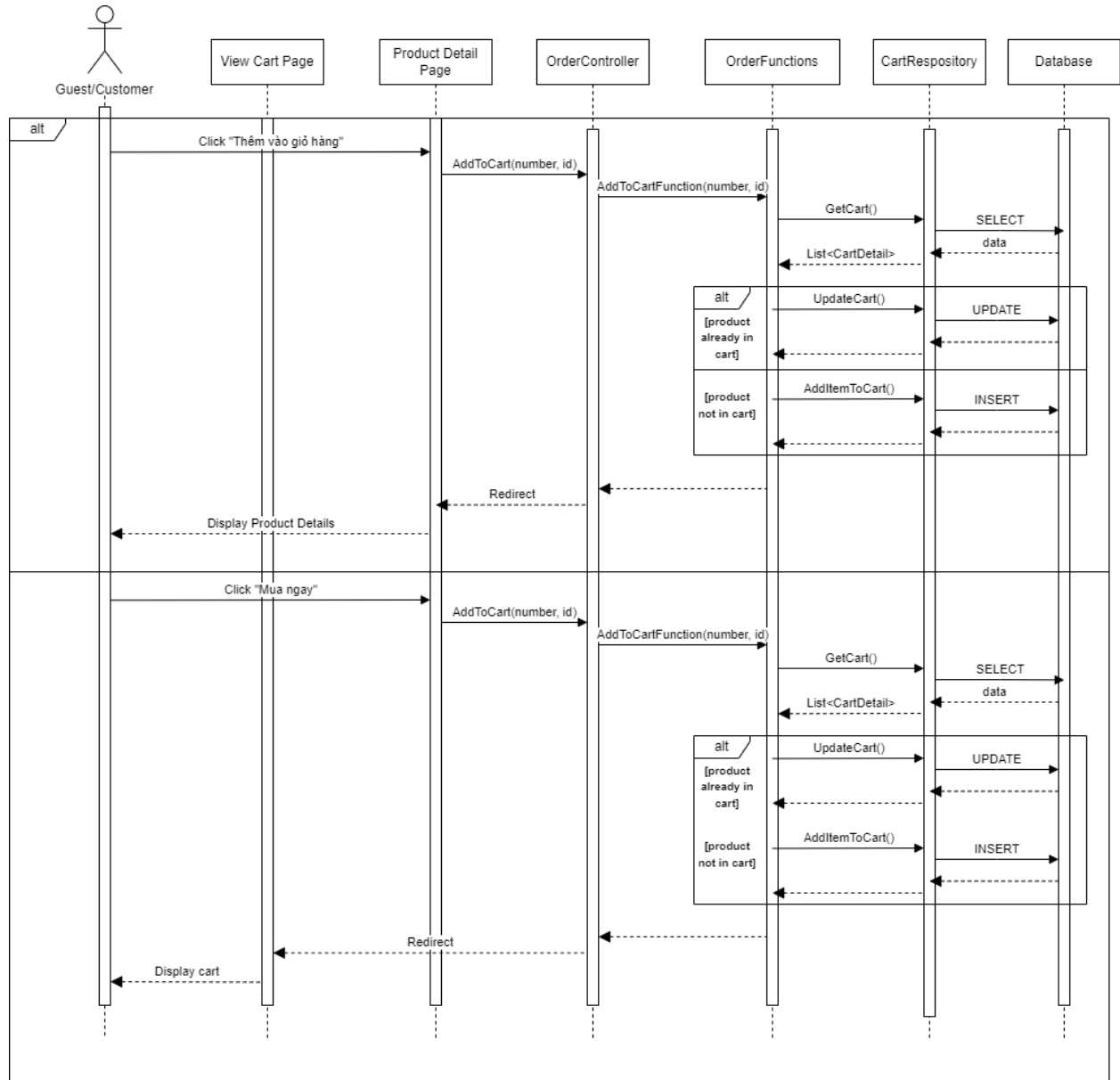
Figure 3.5.4.1: Add Product to Cart functions sequence diagram

3.5.5 View Cart
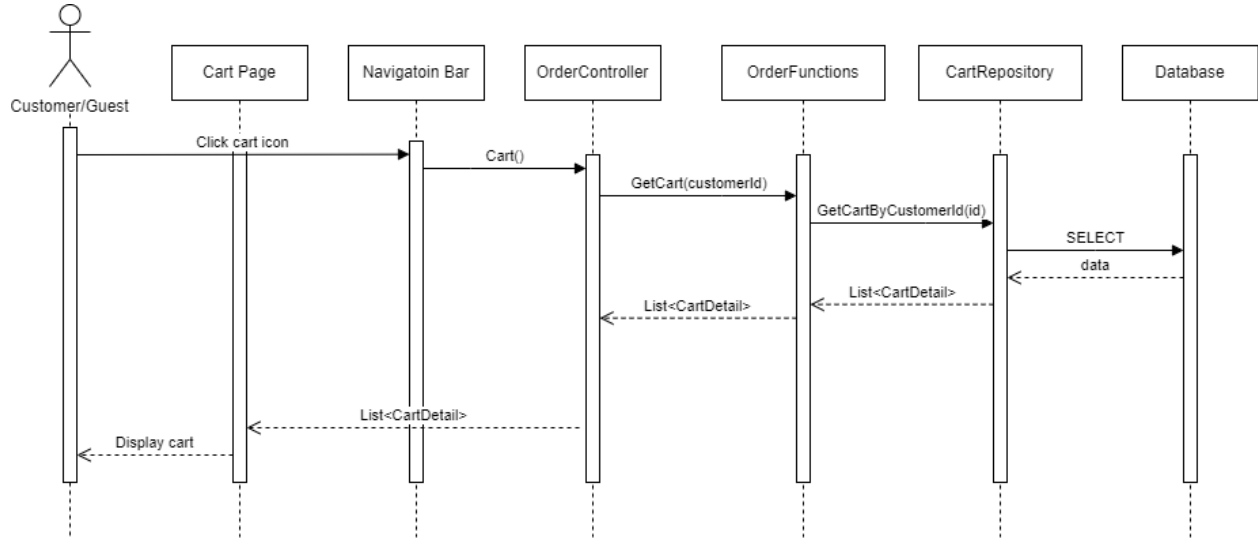
Figure 3.5.5.1: View Cart functions sequence diagram

3.5.6 Update Cart
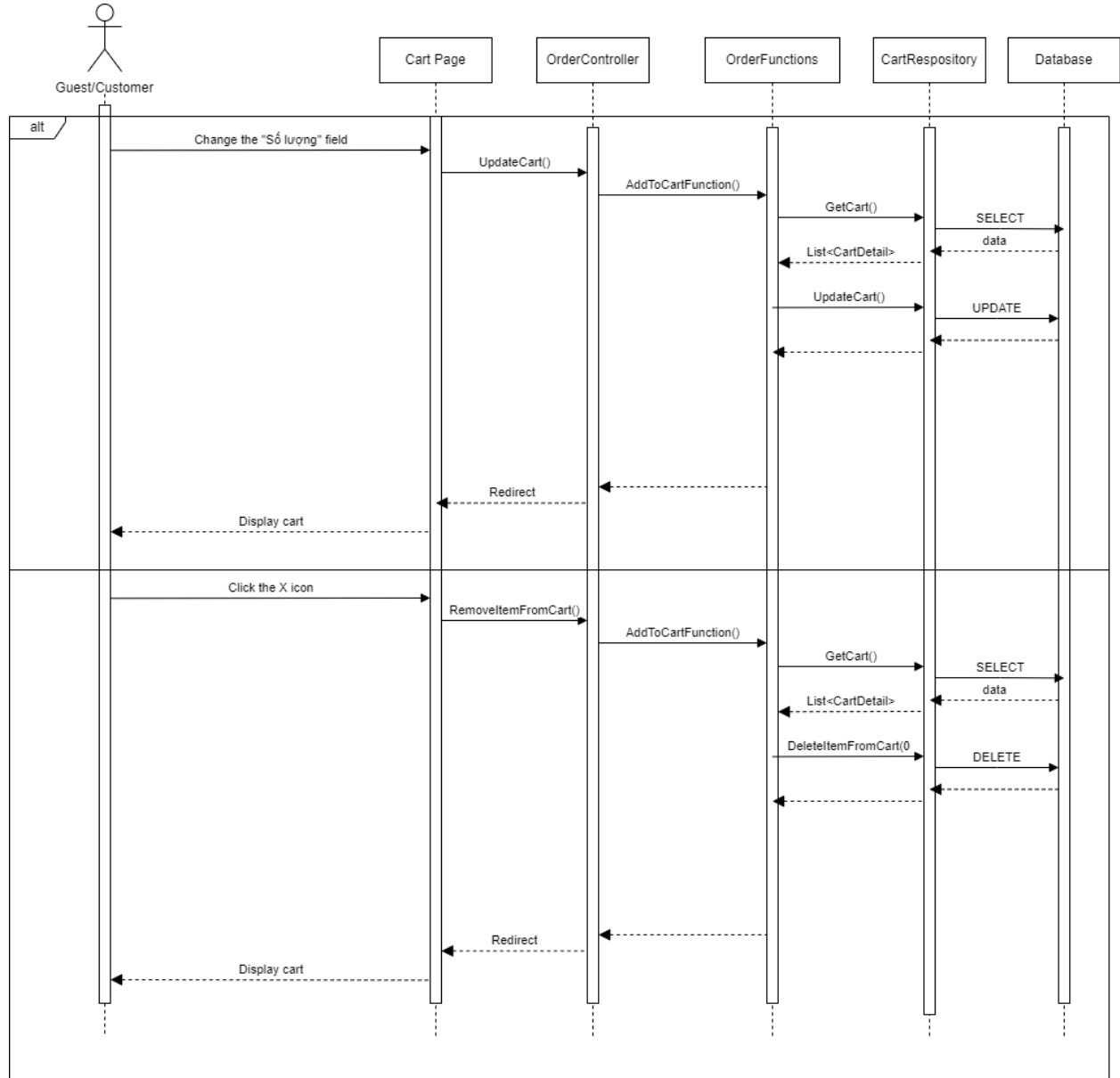
Figure 3.5.6.1: Update Cart functions sequence diagram

3.5.7 Make an Order (Customer)
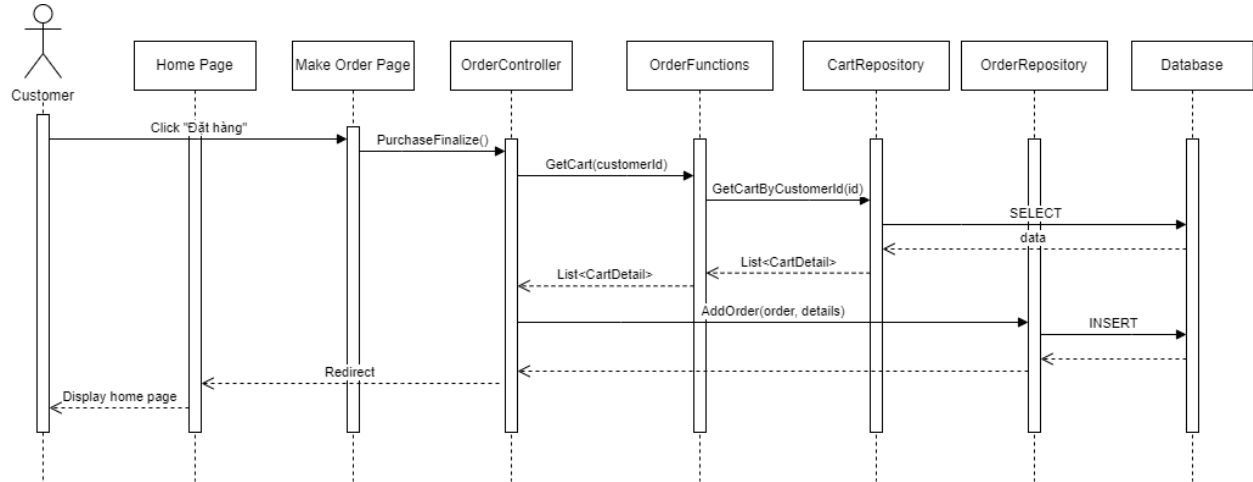
Figure 3.5.6.1: Update Cart functions sequence diagram

3.6 Order Management
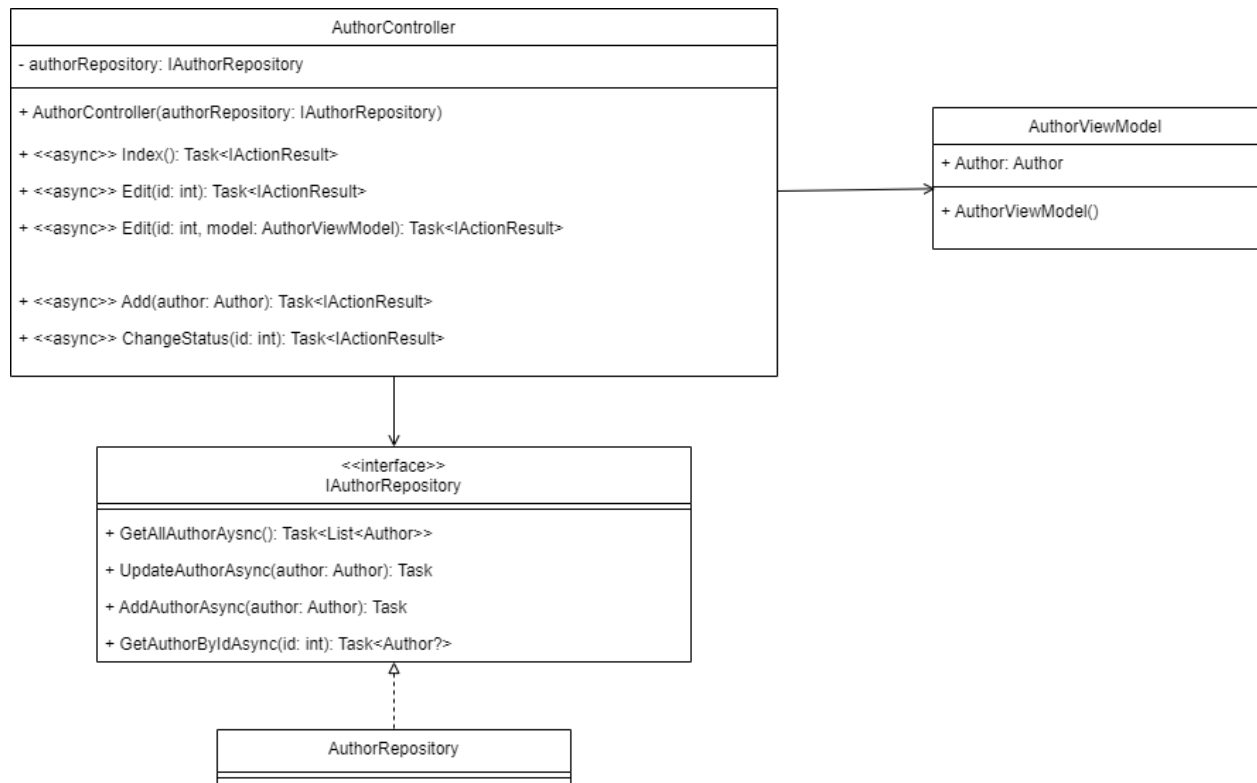
3.6.1 Class Diagram
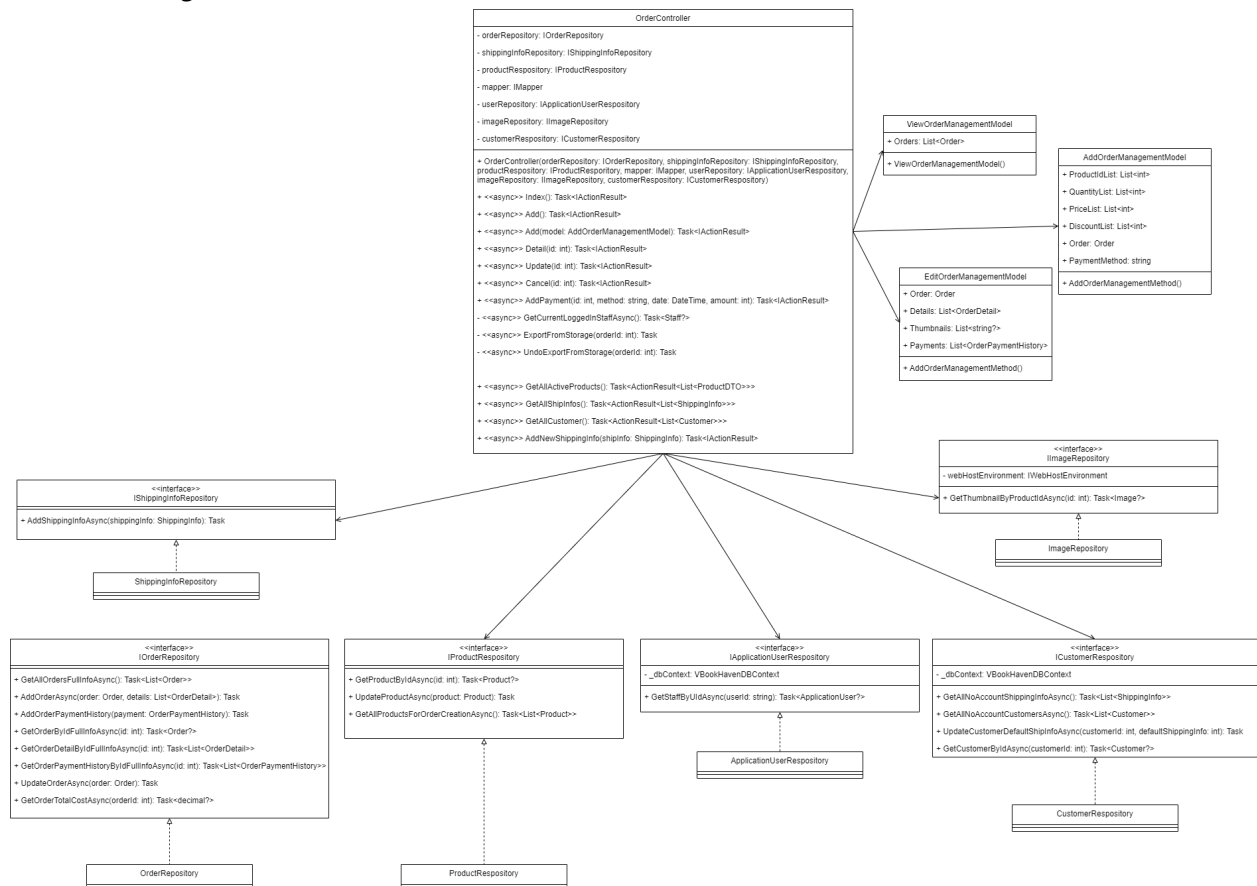
*Figure 3.6.1.1: Order Management functions class diagram*

3.6.2 View List Orders
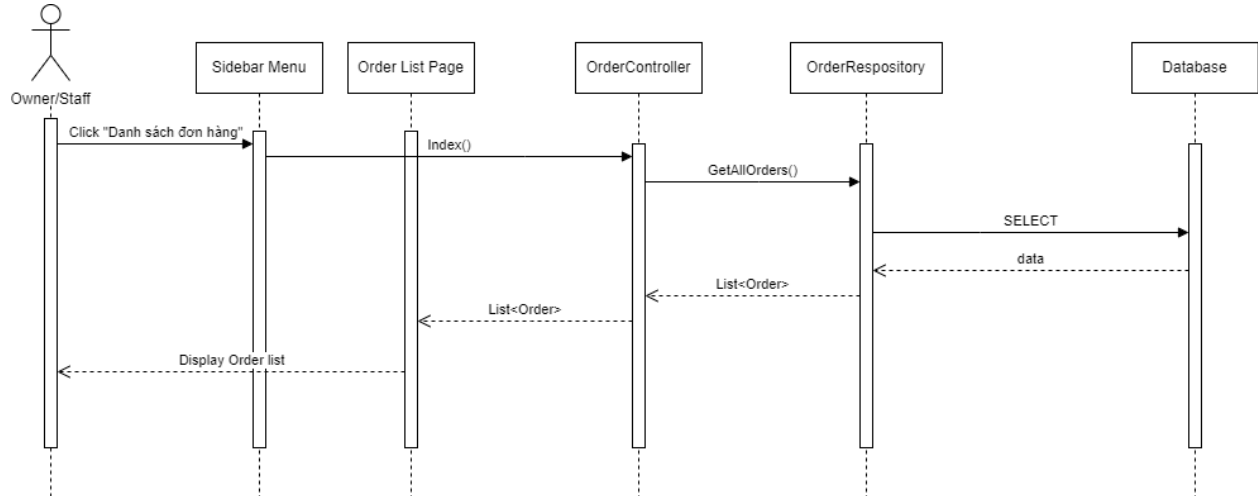
Figure 3.6.2.1: View List Orders functions sequence diagram

3.6.3 Create Order
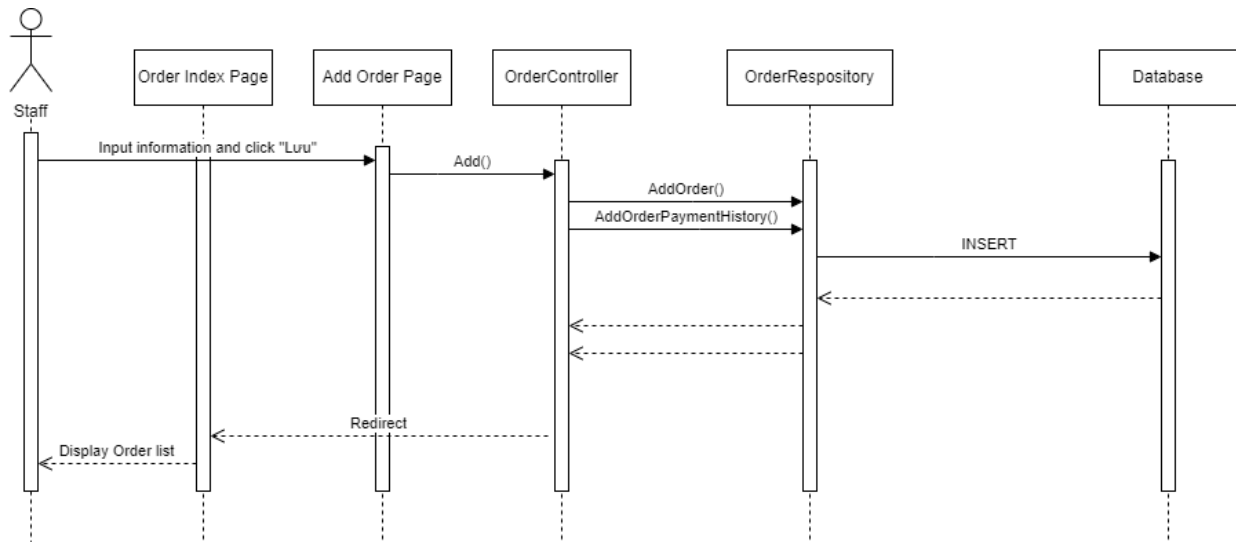
Figure 3.6.3.1: Create Order functions sequence diagram

3.6.4 View Order Details

Figure 3.6.4.1: View order details sequence diagram

3.6.5 Update Order Status

Figure 3.6.5.1: Update Order Status sequence diagram

3.6.6 Add Order Payment

Figure 3.6.6.1: Add Order Payment sequence diagram

3.7 Purchase Order Management

3.7.1 Class Diagram

PurchaseController

IApplicationUserRespository _IApplicationUserRespository;

IProductRespository _productRespository;

IPurchaseOrderRepository _purchaseOrderRepository;

IMapper _mapper;

public PurchaseOrderController(IMapper mapper,
IApplicationUserRespository applicationUserRespository,
, IProductRespository productRespository
, IPurchaseOrderRepository purchaseOrderRepository)

+ <<async>> Index(): Task<IActionResult>

+ <<async>> Create(): Task<IActionResult>

+ <<async>> Create(CreatePurchaseOrderVM model) Task<IActionResult>

+ <<async>> Edit(int purchaseId) Task<IActionResult>

+ <<async>> Edit(CreatePurchaseOrderVM model): Task<IActionResult>

+ <<async>>GetStaffByUserID(): Task<IActionResult>

+ <<async>> Details(int? id): Task<IActionResult>

+ <<async>> UpdateStatus(int id): Task<IActionResult>

+ <<async>> CancelPurchaseOrder(int id): Task<IActionResult>

+ <<async>> ImportToStorage(): Task

+ <<async>> PurchasePayment(DetailsPurchaseOrderVM model): Task<IActionResult>

+ <<async>> totalPaymentNotVAT(ICollection<PurchaseOrderDetail> purchaseOrderDetails): Task<IActionResult>

+ <<async>>totalPaid(ICollection<PurchasePaymentHistory> purchasePaymentHistories): decimal?

DetailsPurchaseOrderVM

+ po: PurchaseOrder

+ TotalPayment: decimal

+ Paid: decimal

+ Unpaid: decimal

+ PPHistory: PurchasePaymentHistory

+ DetailsPurchaseOrderVM()

CreatePurchaseOrderVM

+ SupplierID: int

+ Note: string

+ ProductIdList: List<int>

+ QuantityList: List<int>

+ DiscountList: List<double>

+ AmountPaid: decimal?

+ VAT: double?

+ PurchaseOrderEdit: PurchaseOrder?

+ TotalPayment: decimal?
+ TotalPaid: decimal?
+ DefaultSupplierIDShow: int?

+ CreatePurchaseOrderVM()

<<interface>>
IApplicationUserRespository

+ GetStaffByUIdAsync(String userId): Task<ApplicationUser?>

+ GetAllStaffAsync(): Task<List<ApplicationUser?>>

+ UpdateStaffByAsync(ApplicationUser applicationUser): Task

+ GetCustomerByUIdAsync(String userId): Task<ApplicationUser?>

+ GetAllCustomersAsync(): Task<List<Customer>>

ApplicationUserRespository

<<interface>>
IProductRespository

+ GetAllProductAsync(): Task<List<Product>>

+ AddBookAsync(product: Product, book: Book): Task

+ AddAuthorsToBookAsync(productId: int, AuthorIdList: List<int>): Task

+ AddStationeryAsync(product: Product, stationery: Stationery): Task

+ GetProductByIdAsync(id: int): Task<Product?>

+ GetBookByIdAsync(id: int): Task<Book?>

+ GetStationeryByIdAsync(id: int): Task<Stationery?>

+ GetProductMoreInfoByIdAsync(id: int): Task<Product?>

+ GetBookMoreInfoByIdAsync(id: int): Task<Book?>

+ UpdateProductAsync(product: Product): Task

+ UpdateBookAsync(book: Book): Task

+ UpdateBookAuthorAsync(bookId: int, AuthorIdList: List<int>): Task

+ UpdateStationeryAsync(stationery: Stationery): Task

+ ChangeStatusProductAsync(id: int): Task

+ GetProductByBarcodeAsync(barcode: string): Task<Product?>

ProductRespository

<<interface>>
purchaseOrderRepository

+ AddPurchaseOrderAsync(PurchaseOrder purchaseOrder): Task

+ UpdatePurchaseOrderAsync(PurchaseOrder purchaseOrder): Task

+ UpdatePurchaseOrderStatusAsync(PurchaseOrder purchaseOrder): Task

+ GetPurchaseOrderDetailsAsync(int id): Task<PurchaseOrder?>

PurchaseRepository

Figure 3.7.1.1: Purchase Order Management class diagram

3.7.2 View List Purchase Orders

Figure 3.7.2.1: View List Purchase Orders sequence diagram

3.7.3 Update Purchase Orders Status

Figure 3.7.3.1: Update Purchase Orders Status sequence diagram

3.7.4 Create Purchase Orders

Figure 3.7.4.1: Create Purchase Orders sequence diagram

3.7.5 Update purchase order

Figure 3.7.5.1: Update purchase order sequence diagram

3.8 Account Staff Management

3.8.1 Class Diagram

UserController

- _signInManager: SignInManager<IdentityUser>
- _userManager: UserManager<IdentityUser>
- _roleManager: RoleManager<IdentityRole>
- _userStore: IUserStore<IdentityUser>
- _emailSender: IEmailSender
- _webHostEnvironment: IWebHostEnvironment
- _IApplicationUserRespository: IApplicationUserRespository
- _emailSender: IEmailSender

---

public UserController(
      UserManager<IdentityUser> userManager,
      RoleManager<IdentityRole> roleManager,
      IUserStore<IdentityUser> userStore,
      SignInManager<IdentityUser> signInManager,
      IEmailSender emailSender,
      IApplicationUserRespository applicationUserRespository,
      IWebHostEnvironment webHostEnvironment)

+ <<async>> Create(): Task<IActionResult>
+ <<async>> Create(UserVM model, string? returnUrl): Task<IActionResult>
+ <<async>> Profile(): Task<IActionResult>
+ <<async>>Profile(ProfileVM profileVM): Task<IActionResult>
+ <<async>> ChangePassword(ProfileVM profileVM): Task<IActionResult>
+ <<async>> CreateUser(): ApplicationUser
+ <<async>> GetRoleListToUpdateStaff(String selectedRole): IEnumerable<SelectListItem>
+ <<async>> GetEmailStore(): IUserEmailStore<IdentityUser>
+ <<async>> EditStationery(id: int, model: ProductManagementViewModel): Task<IActionResult>
+ <<async>> Index(): Task<IActionResult>
+ <<async>> Edit(String userId): Task<IActionResult>
+ <<async>> Edit(UpdateStaffVM model, string? returnUrl): Task<IActionResult>
+ <<async>> updateRoleByUIDAsync(string uid, string newRole): Task<ActionResult<string>>

ProfileVM

+ Product: Product
+ Book: Book
+ Stationery: Stationery
+ AuthorIdList: List<int>
+ AddImageList: List<IFormFile>
+ DeleteImageIdList: List<int>

+ ProductManagementViewModel()

StaffProfileVM

+ Email: Product
+ Password: string
+ Role: string
+RoleList: IEnumerable<SelectListItem>
+ ApplicationUser: ApplicationUser
+ Staff_ImageFile: IFormFile

+StaffProfileVM()

ChangePwdVM

+ CurrentPwd: string
+ NewPwd: string
+ ConfirmNewPwd: string

+ ChangePwdVM()

**<<interface>>
_IApplicationUserRespository**

+ GetStaffByUIdAsync(String userId): Task<ApplicationUser?>

GetAllStaffAsync(): Task<List<ApplicationUser?>>

UpdateStaffByAsync(ApplicationUser applicationUser): Task

UpdateStaffByAsync(ApplicationUser applicationUser): Task

GetCustomerByUIdAsync(String userId): Task<ApplicationUser?>

GetAllCustomersAsync(): Task<List<Customer>>

ApplicationUserRespository

Figure 3.8.1.1: Account Staff Management class diagram

3.8.2 View List Account Staff

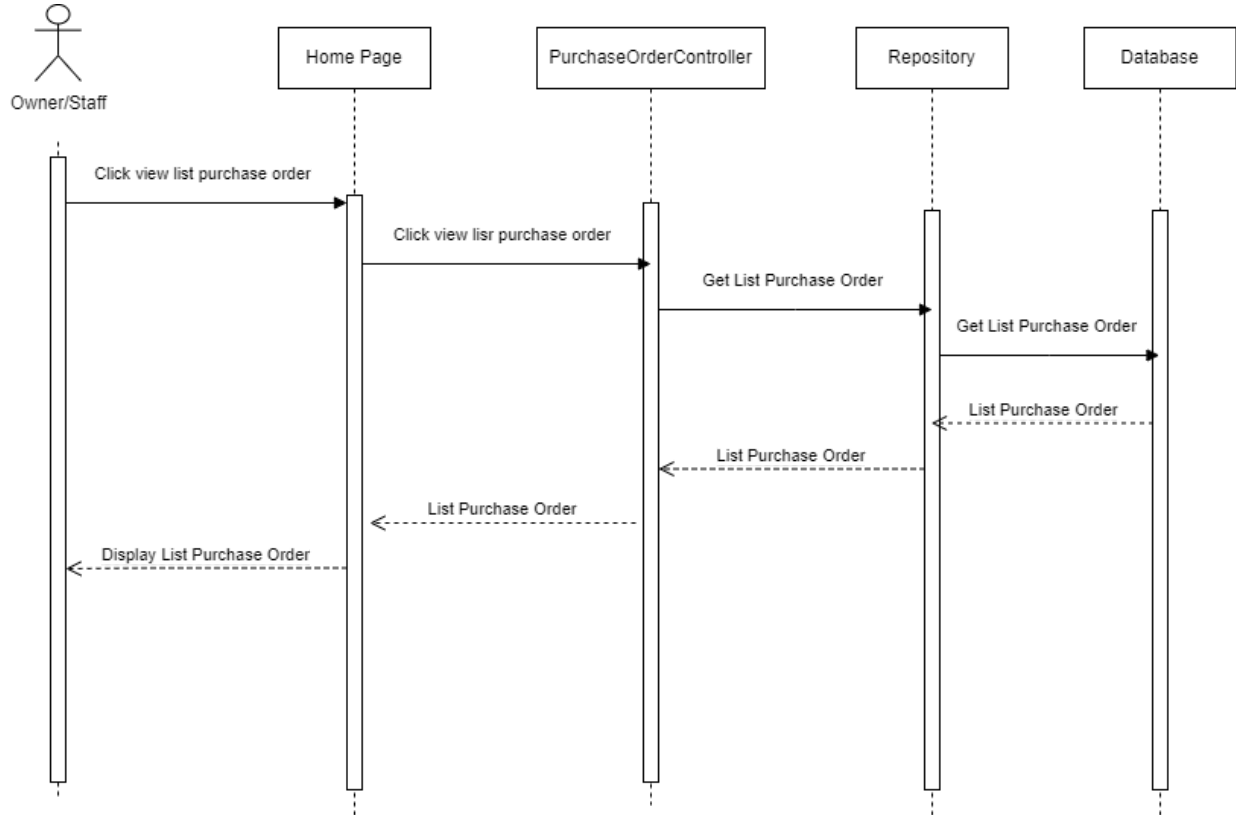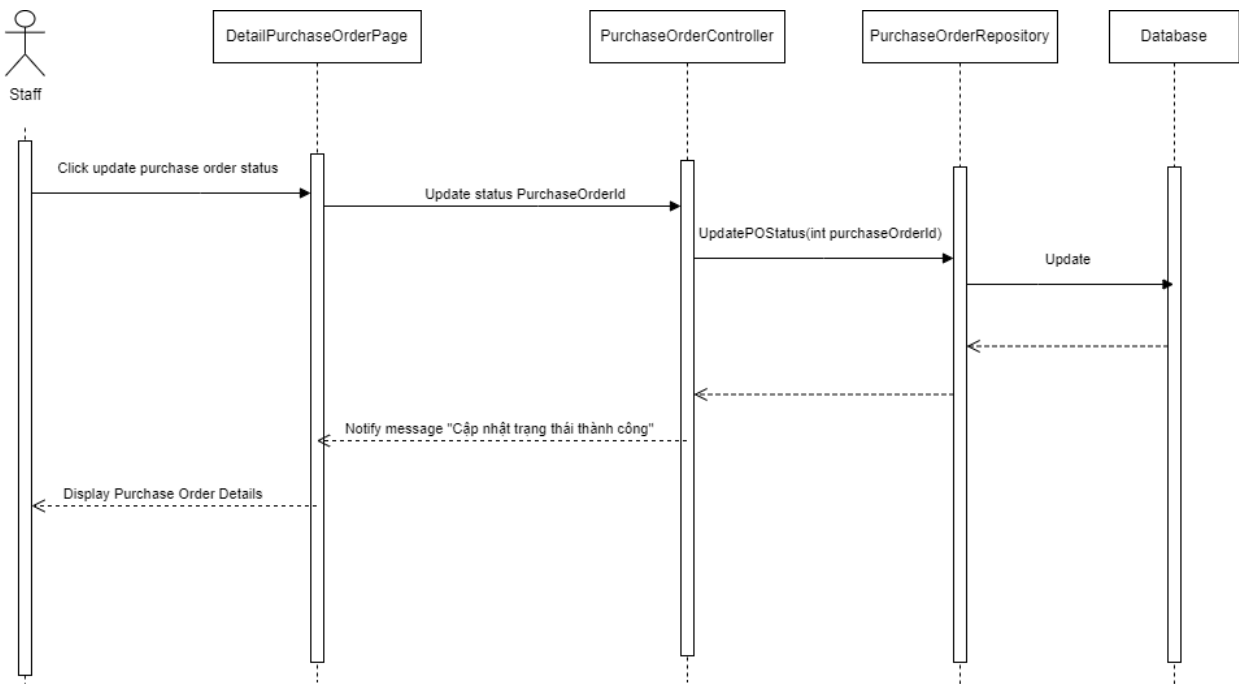*Figure 3.8.2.1: View List Account Staff sequence diagram*

3.8.3 Create Account

Figure 3.8.3.1: Create Account sequence diagram

3.8.4 Update Account

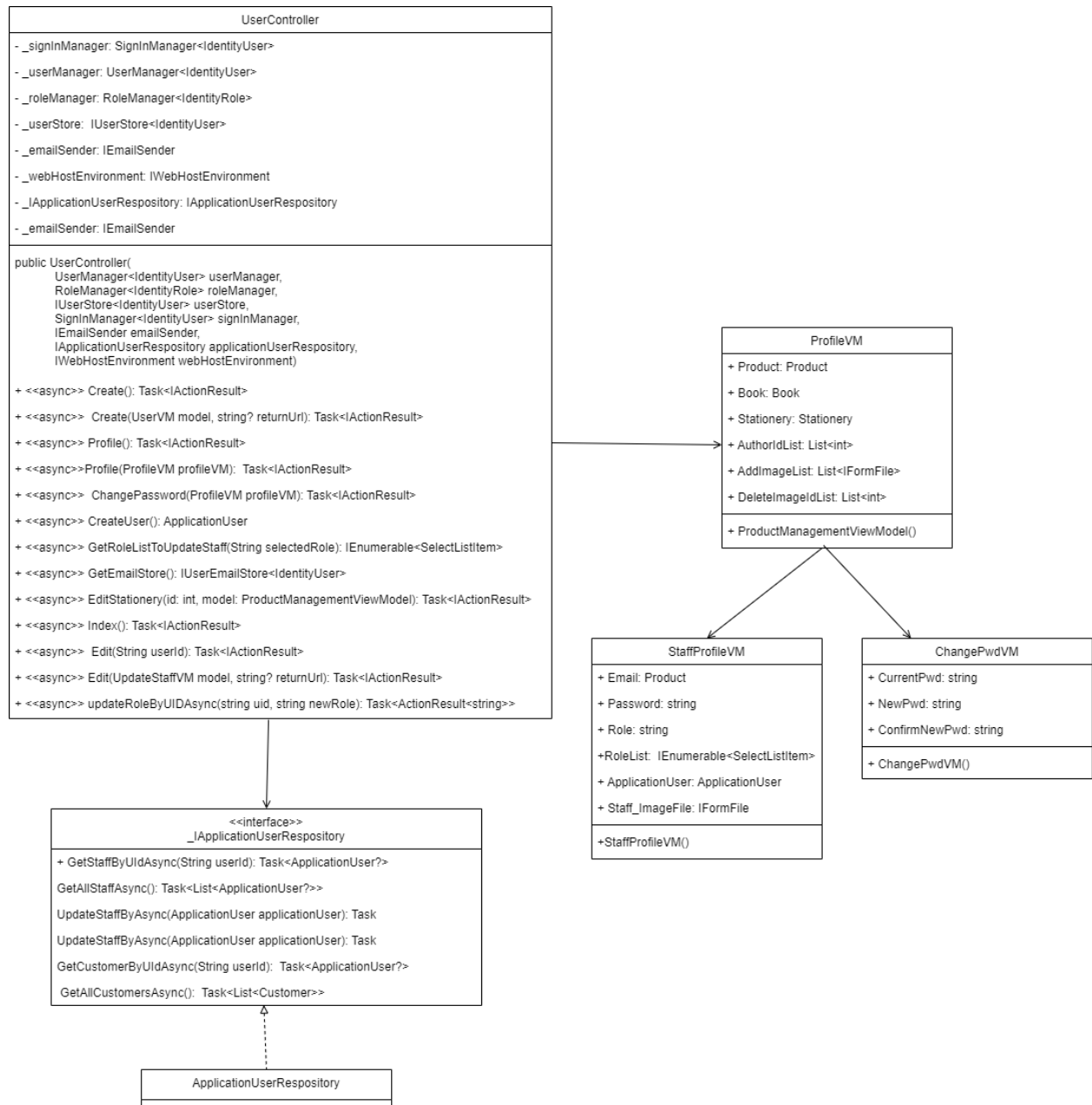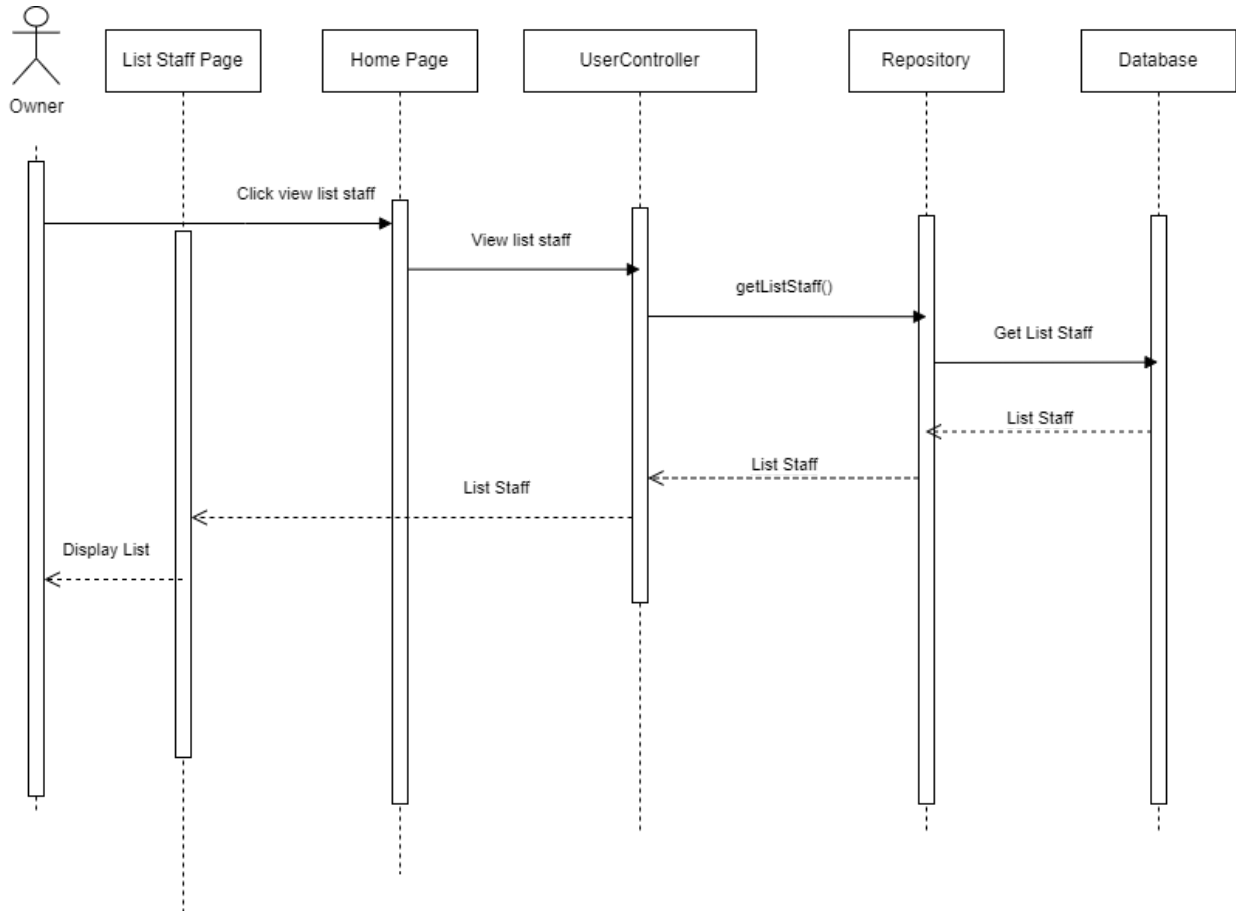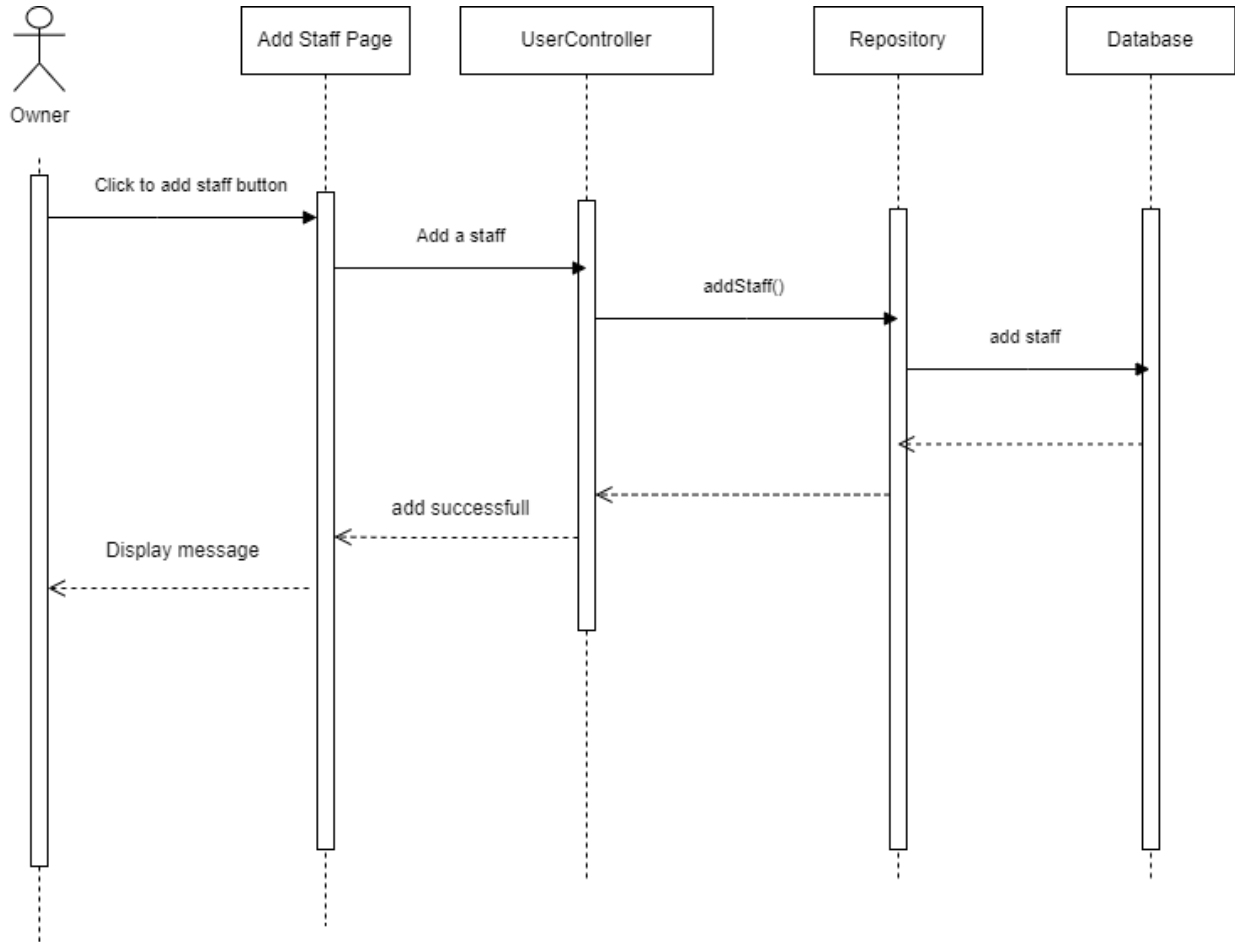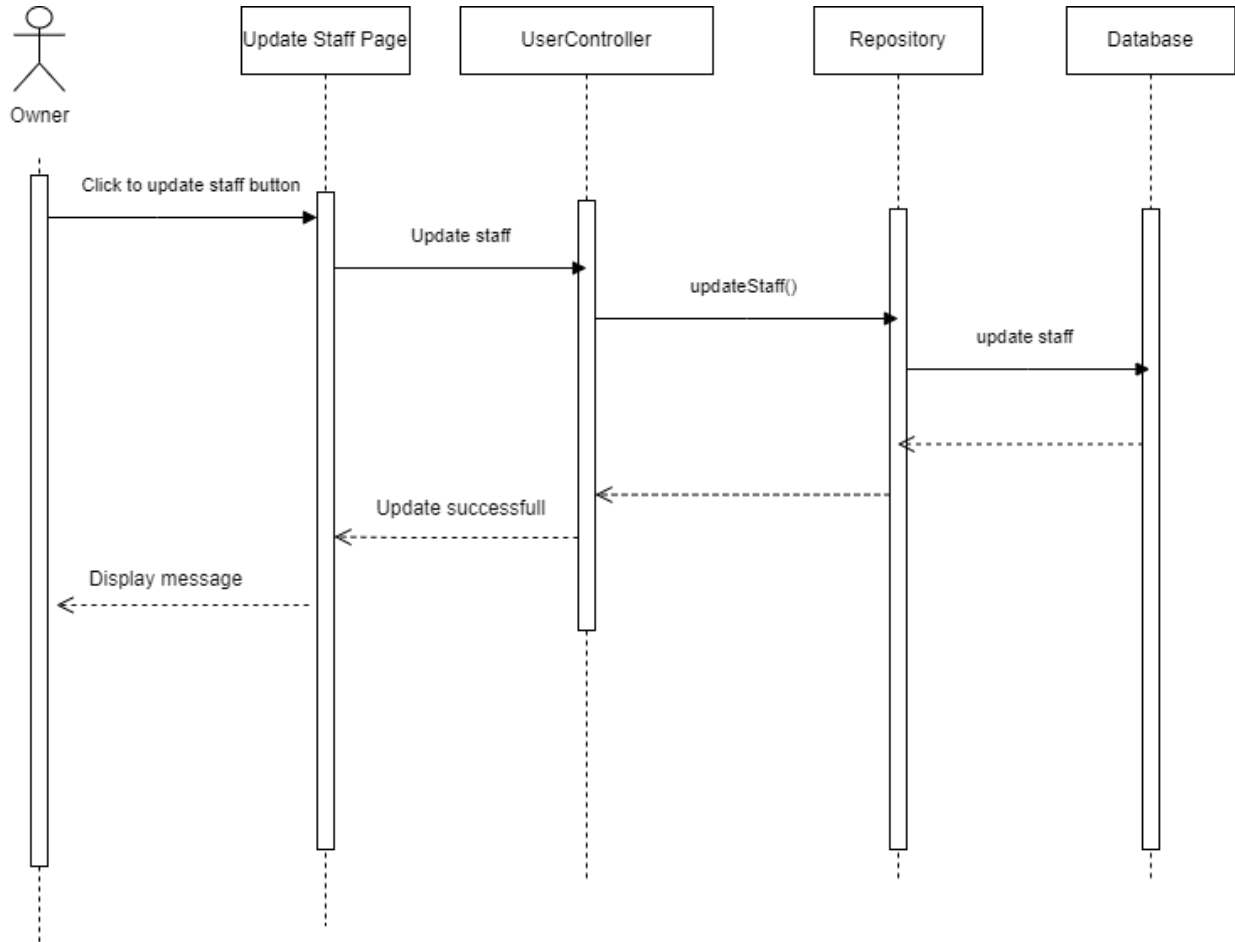*Figure 3.8.4.1: Update Account sequence diagram*
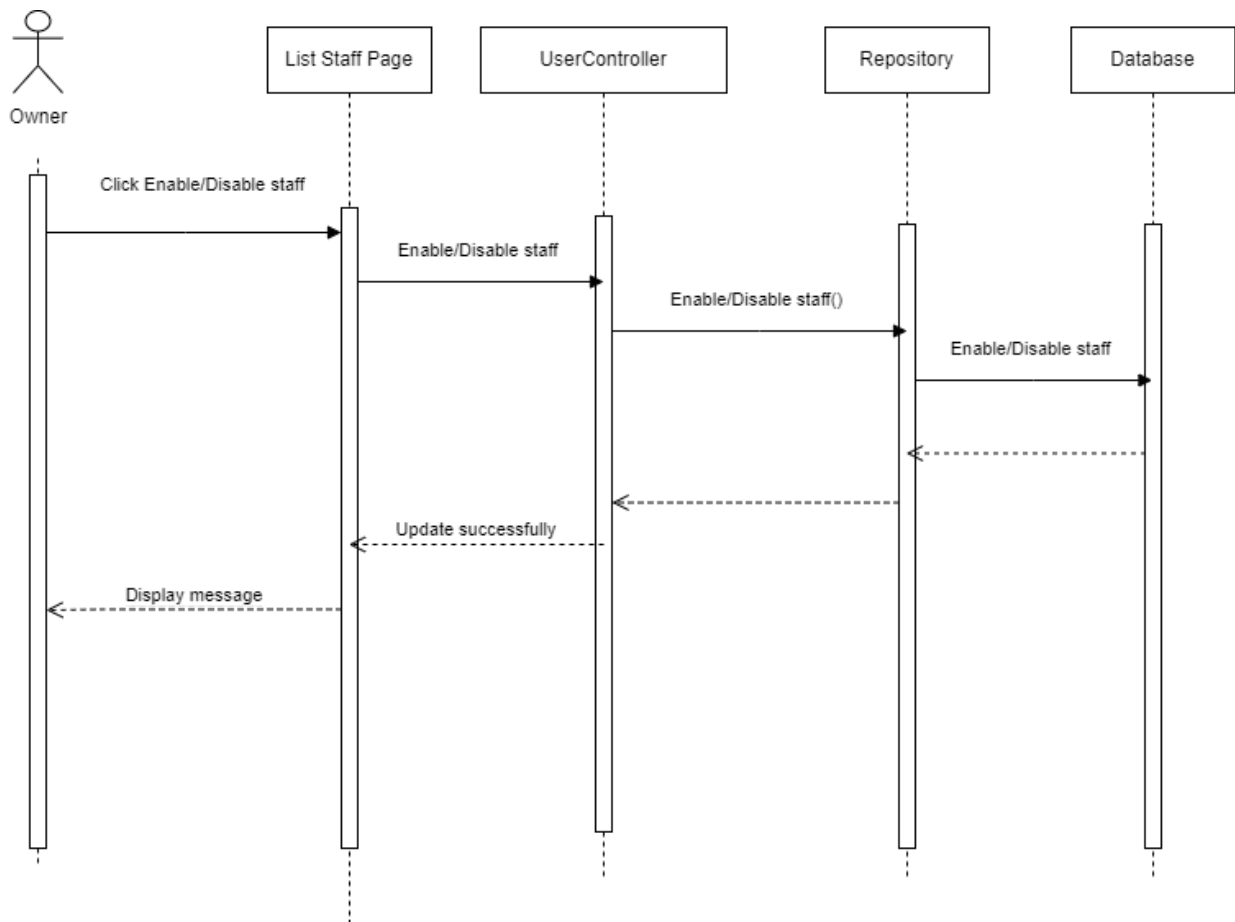
3.8.5 Enable/Disable staff

Figure 3.8.5.1: Enable/Disable staff sequence diagram

3.9 (Staff/Owner) Profile Management
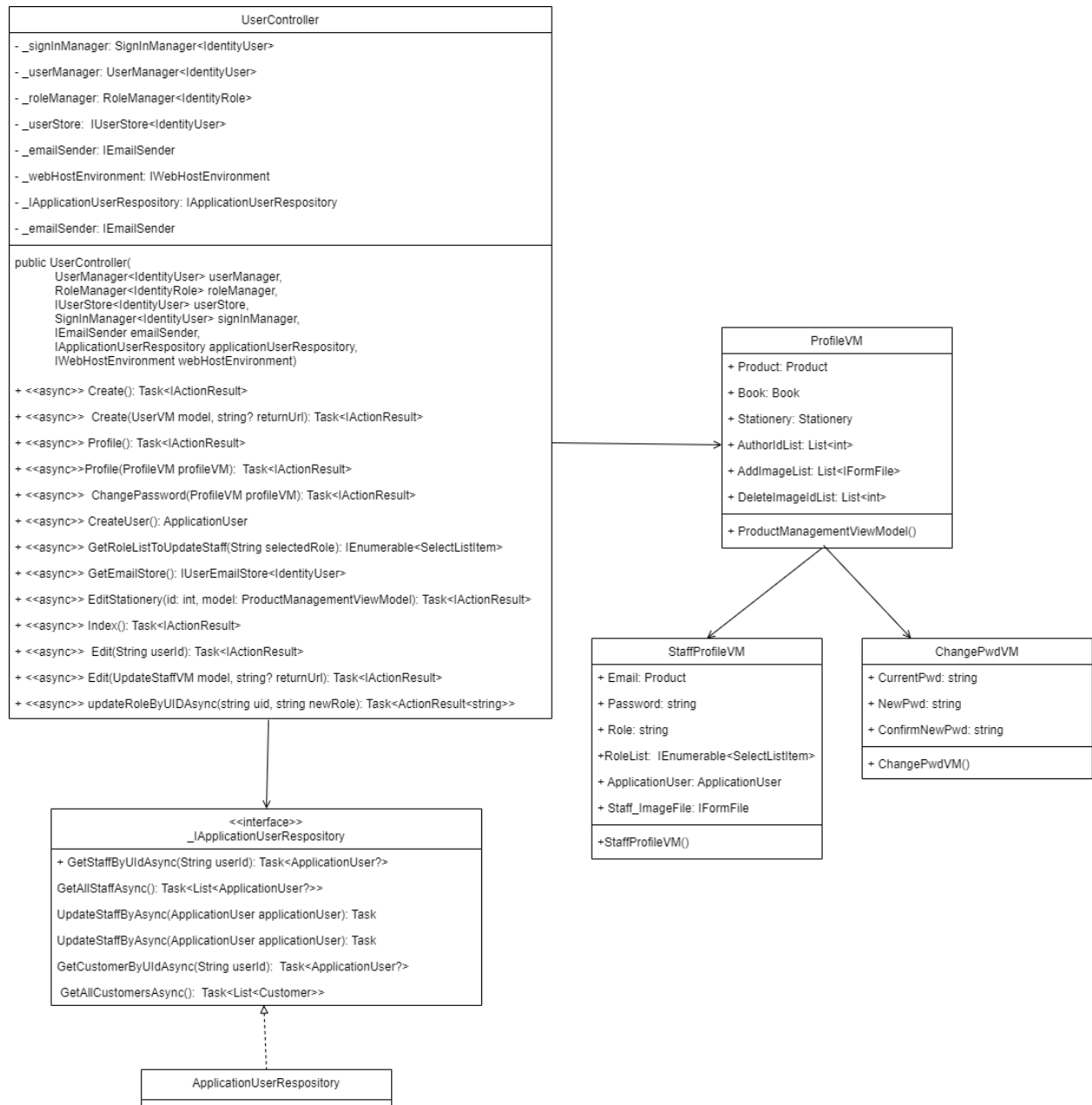
3.9.1 Class Diagram

UserController
- _signInManager: SignInManager<IdentityUser>
- _userManager: UserManager<IdentityUser>
- _roleManager: RoleManager<IdentityRole>
- _userStore: IUserStore<IdentityUser>
- _emailSender: IEmailSender
- _webHostEnvironment: IWebHostEnvironment
- _IApplicationUserRespository: IApplicationUserRespository
- _emailSender: IEmailSender

public UserController(
        UserManager<IdentityUser> userManager,
        RoleManager<IdentityRole> roleManager,
        IUserStore<IdentityUser> userStore,
        SignInManager<IdentityUser> signInManager,
        IEmailSender emailSender,
        IApplicationUserRespository applicationUserRespository,
        IWebHostEnvironment webHostEnvironment)

+ <<async>> Create(): Task<IActionResult>
+ <<async>> Create(UserVM model, string? returnUrl): Task<IActionResult>
+ <<async>> Profile(): Task<IActionResult>
+ <<async>>Profile(ProfileVM profileVM): Task<IActionResult>
+ <<async>> ChangePassword(ProfileVM profileVM): Task<IActionResult>
+ <<async>> CreateUser(): ApplicationUser
+ <<async>> GetRoleListToUpdateStaff(String selectedRole): IEnumerable<SelectListItem>
+ <<async>> GetEmailStore(): IUserEmailStore<IdentityUser>
+ <<async>> EditStationery(id: int, model: ProductManagementViewModel): Task<IActionResult>
+ <<async>> Index(): Task<IActionResult>
+ <<async>> Edit(String userId): Task<IActionResult>
+ <<async>> Edit(UpdateStaffVM model, string? returnUrl): Task<IActionResult>
+ <<async>> updateRoleByUIDAsync(string uid, string newRole): Task<ActionResult<string>>

ProfileVM
+ Product: Product
+ Book: Book
+ Stationery: Stationery
+ AuthorIdList: List<int>
+ AddImageList: List<IFormFile>
+ DeleteImageIdList: List<int>
+ ProductManagementViewModel()

StaffProfileVM
+ Email: Product
+ Password: string
+ Role: string
+RoleList: IEnumerable<SelectListItem>
+ ApplicationUser: ApplicationUser
+ Staff_ImageFile: IFormFile
+StaffProfileVM()

ChangePwdVM
+ CurrentPwd: string
+ NewPwd: string
+ ConfirmNewPwd: string
+ ChangePwdVM()

<<interface>> _IApplicationUserRespository
+ GetStaffByUIdAsync(String userId): Task<ApplicationUser?>
GetAllStaffAsync(): Task<List<ApplicationUser?>>
UpdateStaffByAsync(ApplicationUser applicationUser): Task
UpdateStaffByAsync(ApplicationUser applicationUser): Task
GetCustomerByUIdAsync(String userId): Task<ApplicationUser?>
GetAllCustomersAsync(): Task<List<Customer>>

ApplicationUserRespository

Figure 3.9.1.1: (Staff/Owner) Profile Management class diagram
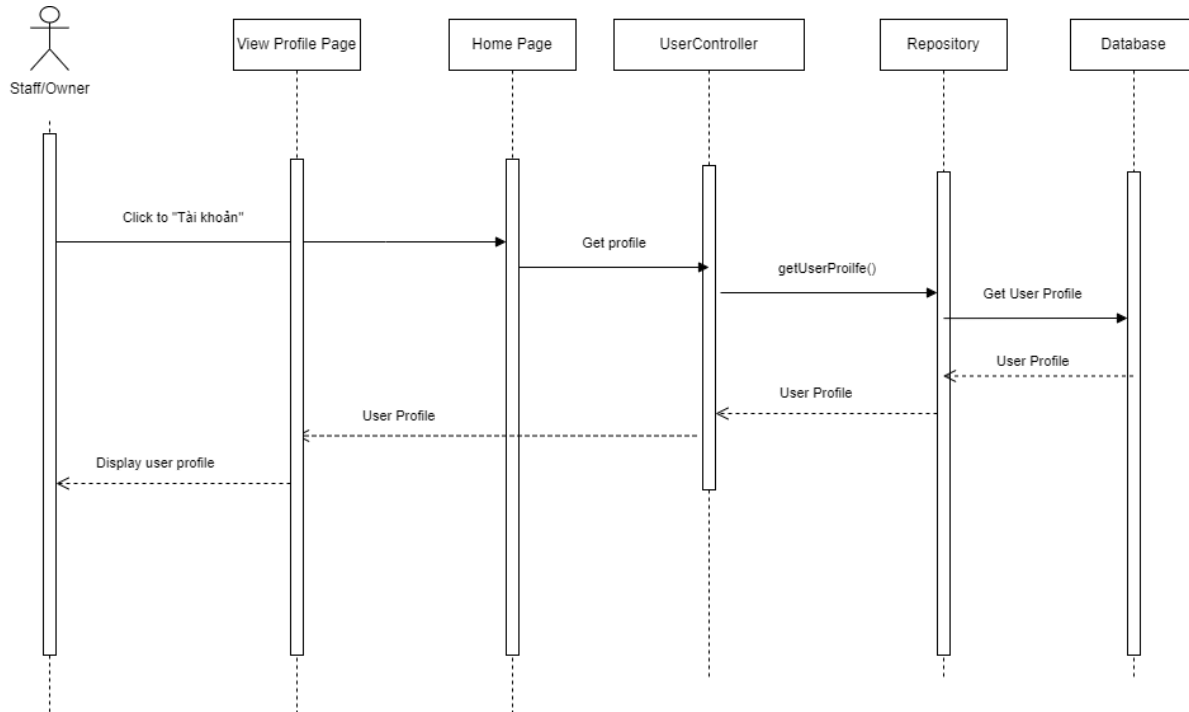
3.9.2 View profile

Figure 3.9.2.1: View profile sequence diagram
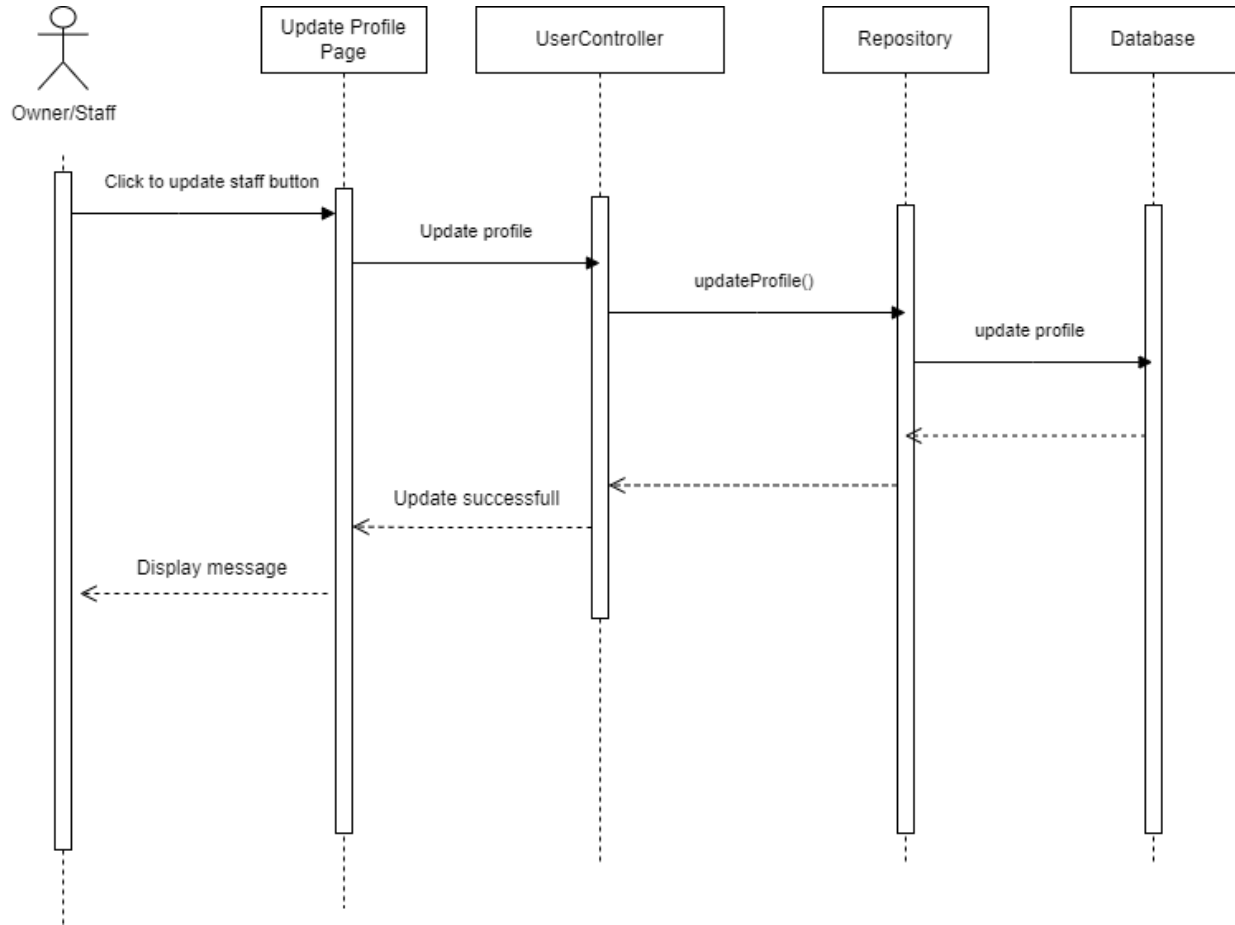
3.9.3 Edit user profile

Figure 3.9.2.1: Edit user profile sequence diagram
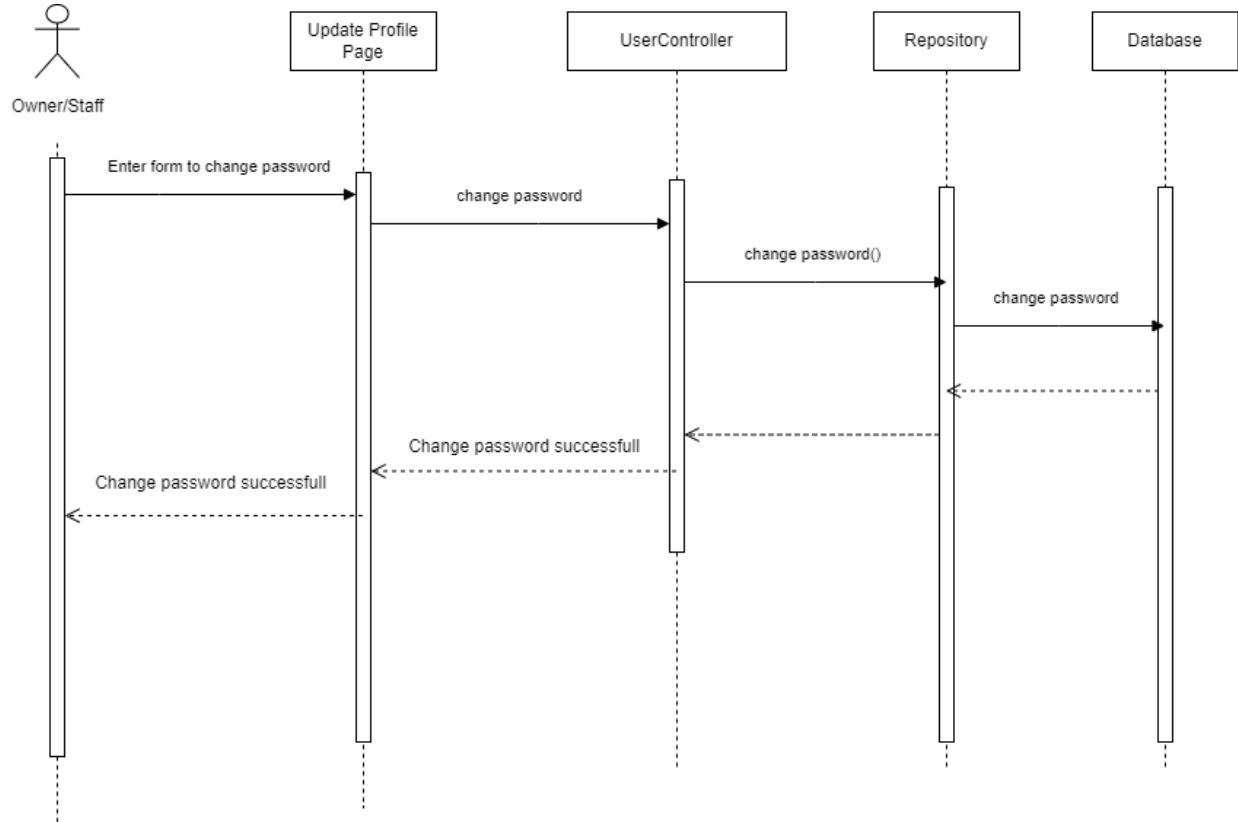
3.9.4 Change password

Figure 3.9.4.1: Change password sequence diagram

3.10 (Customer) Profile Management

3.10.1 Class Diagram

AccountController

- client: HttpClient
- _shippingInfoRepository: UserManager<IdentityUser>
- _customerRespository: ICustomerRespository
- _userManager: UserManager<IdentityUser>
- _applicationUserRespository: IApplicationUserRespository
- _mapper: IMapper
- orderRepository: IOrderRepository
- imageRepository: ImageRepository

AccountController(IShippingInfoRepository shippingInfoRespository,
      UserManager<IdentityUser> userManager,
      IMapper mapper,
      IApplicationUserRespository applicationUserRespository,
      ICustomerRespository customerRespository,
      IOrderRepository orderRepository,
      IImageRepository imageRepository,
      IProductRespository productRespository)

- + <<async>> ShipInfo(): Task<IActionResult>
- + <<async>> CreateShipInfo(): Task<IActionResult>
- + <<async>> EditShipInfo(int shipInfoId): Task<IActionResult>
- + <<async>> EditShipInfo(ShippingInfoVM shippingInfoVM, string? returnUrl): Task<IActionResult>
- + <<async>> CreateShipInfo(ShippingInfoVM shippingInfoVM, string? returnUrl): Task<IActionResult>
- + <<async>> DeleteShipInfo(int shipInfoId): Task<IActionResult>
- + <<async>> Edit(): Task<IActionResult>
- + <<async>> Edit(CustomerProfileVM profileVM, string? returnUrl): Task<IActionResult>
- + <<async>> EditStationery(id: int, model: ProductManagementViewModel): Task<IActionResult>
- + <<async>> ChangePassword(): Task<IActionResult>
- + <<async>> ChangePassword(ChangePwdVM changePwdVM, string? returnUrl): Task<IActionResult>
- + <<async>> getCustomerFromIdentity(): Task<IActionResult>
- + <<async>> updateRoleByUIDAsync(string uid, string newRole): Task<ActionResult<string>>
- + <<async>> Orders(): Task<ActionResult>

CustomerProfileVM

- + Email: string
- + Customer: Customer
- + Customer_ImageFile: IFormFile

- + CustomerProfileVM()

ShippingInfoVM

- + shippingInfosIsNull: bool
- + CustomerId: int
- + IsDefault: bool
- + ShippingInfo: ShippingInfo
- + ShippingInfos: List<ShippingInfo>

- + ShippingInfoVM()

<<interface>> ICustomerRespository

- + UpdateCustomerDefaultShipInfoOnCreateAsync(ShippingInfoVM model)
- + UpdateCustomerDefaultShipInfoAsync(int customerId, int defaultShippingInfoId): Task
- + UpdateCustomerProfile(Customer c): Task
- + AddShippingInfoAsync(ShippingInfo shippingInfo): Task<int>
- + GetAllShipInfoByCusIDAsync(int cusID): Task<List<ShippingInfo>?>
- + GetAllNoAccountCustomersAsync(): Task<List<Customer>>
- + GetAllNoAccountShippingInfoAsync(): Task<List<ShippingInfo>>
- AddCustomerNoAccountAsync(Customer customer): Task

CustomerRespository

<<interface>> IShippingInfoRepository

- + GetAllShipInfoAsync(): Task<List<ShippingInfo>>
- + GetAllShippingInfosByCustomerIdAsync(int customerId): Task<List<ApplicationUser>?>
- + GetShippingInfoByIdAsync(int id): Task<ShippingInfo?>
- + AddShippingInfoAsync(ShippingInfo shippingInfo): Task<int>
- + GetAllShipInfoByCusIDAsync(int cusID): Task<List<ShippingInfo>?>
- GetShipInfoByCusIdAndShipInfoIdAsync(int customerId, int Id:Task<ShippingInfo>)
- UpdateShipInfoAsync(ShippingInfo shippingInfo):Task
- DeleteShipInfoAsync(ShippingInfo shippingInfo):Task

ShippingInfoRepository

<<interface>> IApplicationUserRespository

- + GetStaffByUidAsync(String userId): Task<ApplicationUser?>
- + GetAllStaffAsync(): Task<List<ApplicationUser>>
- + UpdateStaffByAsync(ApplicationUser applicationUser): Task
- + GetCustomerByUIdAsync(String userId): Task<ApplicationUser?>
- + GetAllCustomersAsync(): Task<List<Customer>>
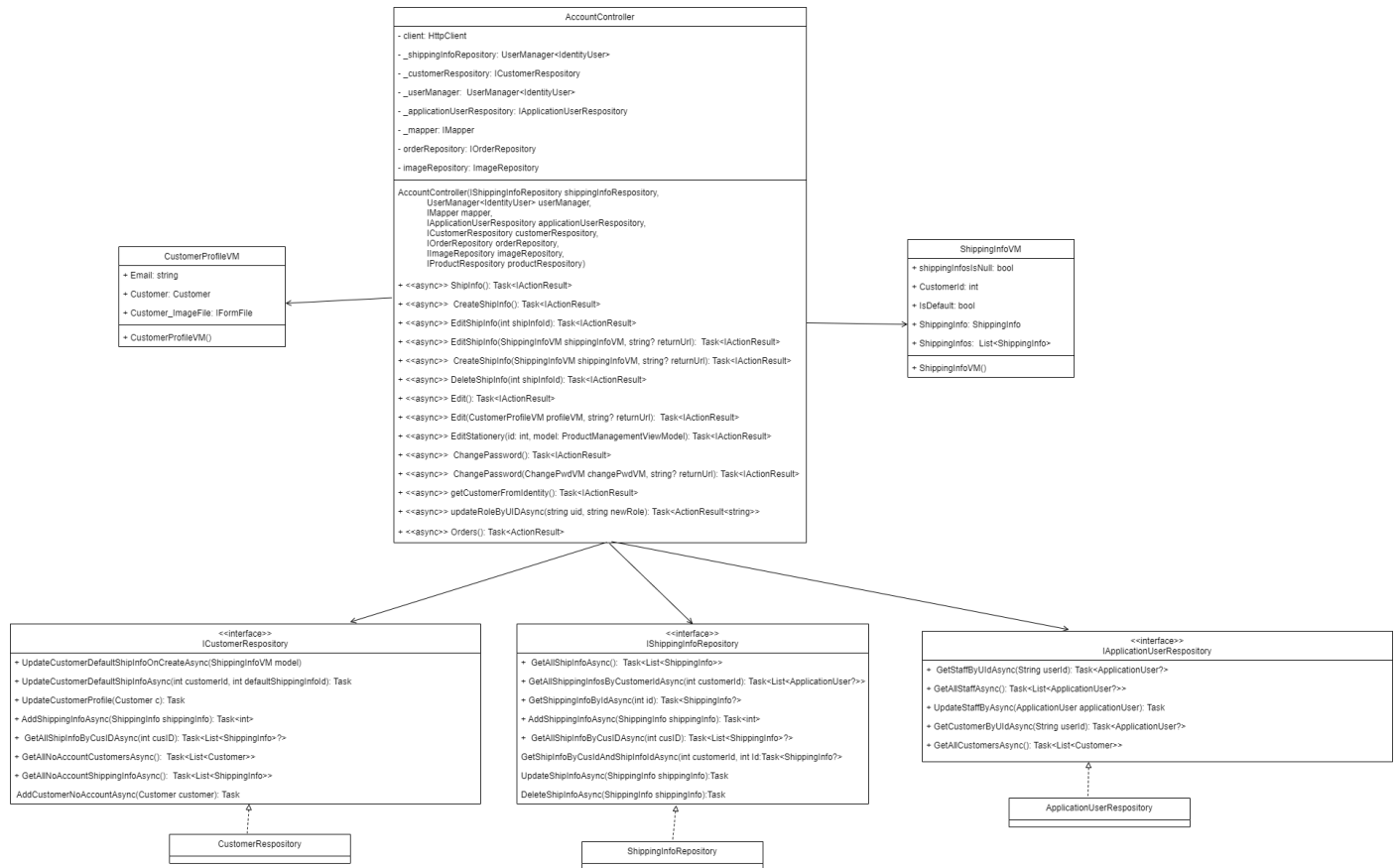
ApplicationUserRespository

Figure 3.10.1.1: (Customer) Profile Management functions class diagram

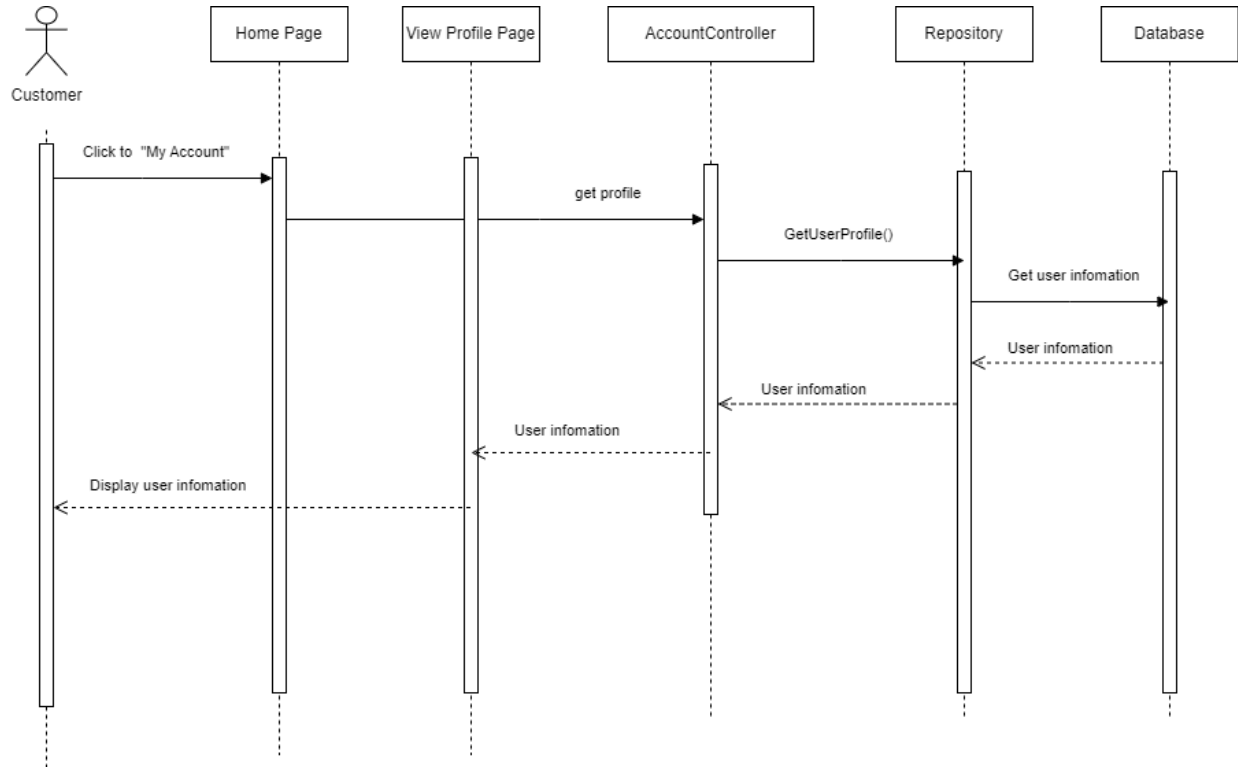### 3.10.2 View profile

Figure 3.10.2.1: View profile sequence diagram
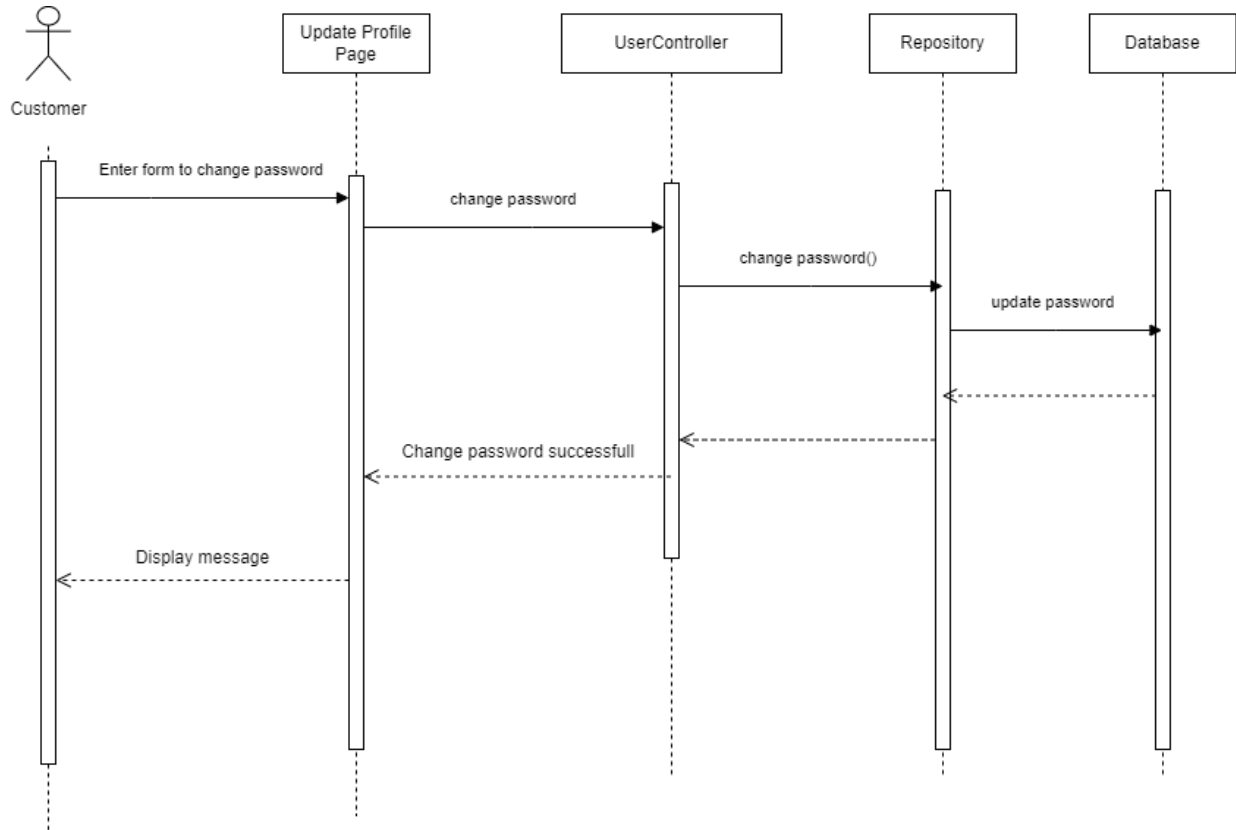
3.10.3 Edit profile

Figure 3.10.3.1: Edit profile sequence diagram
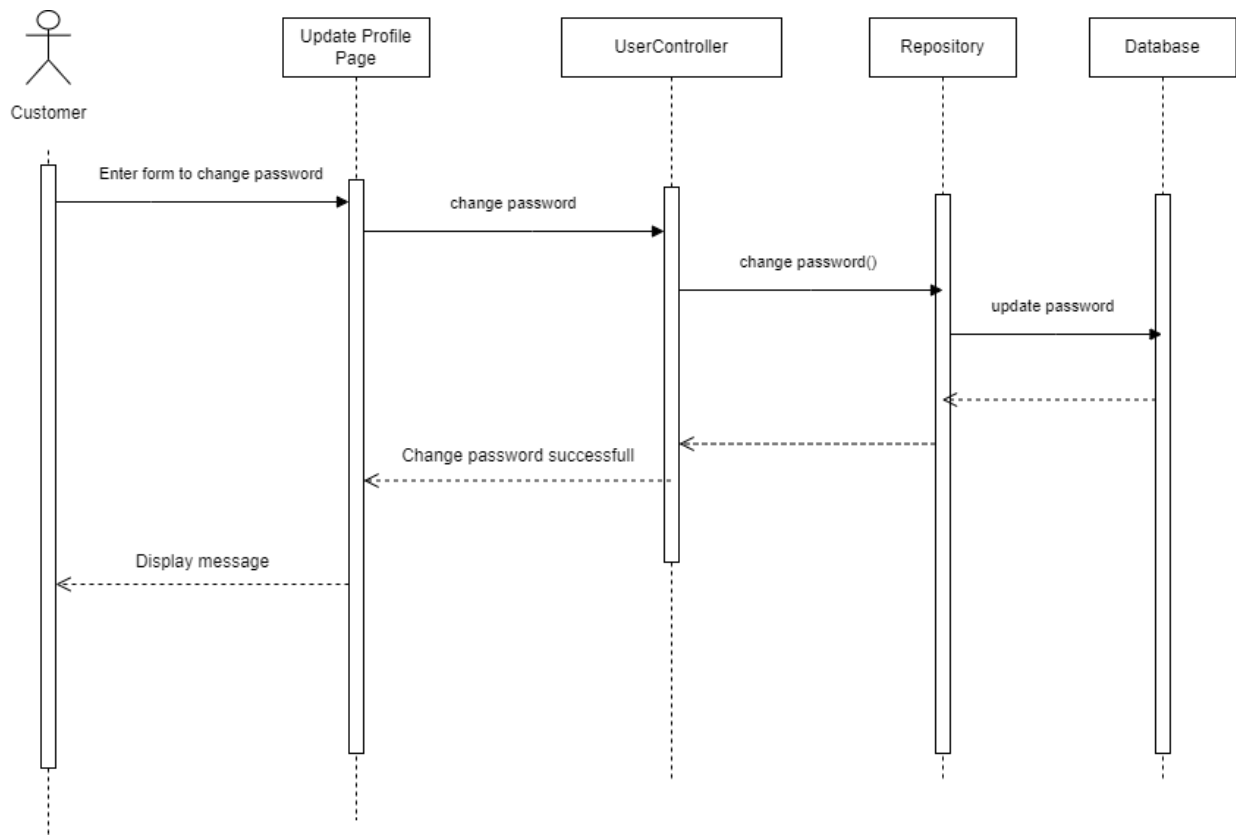
3.10.4 Change Password

Figure 3.10.4.1: Change Password sequence diagram

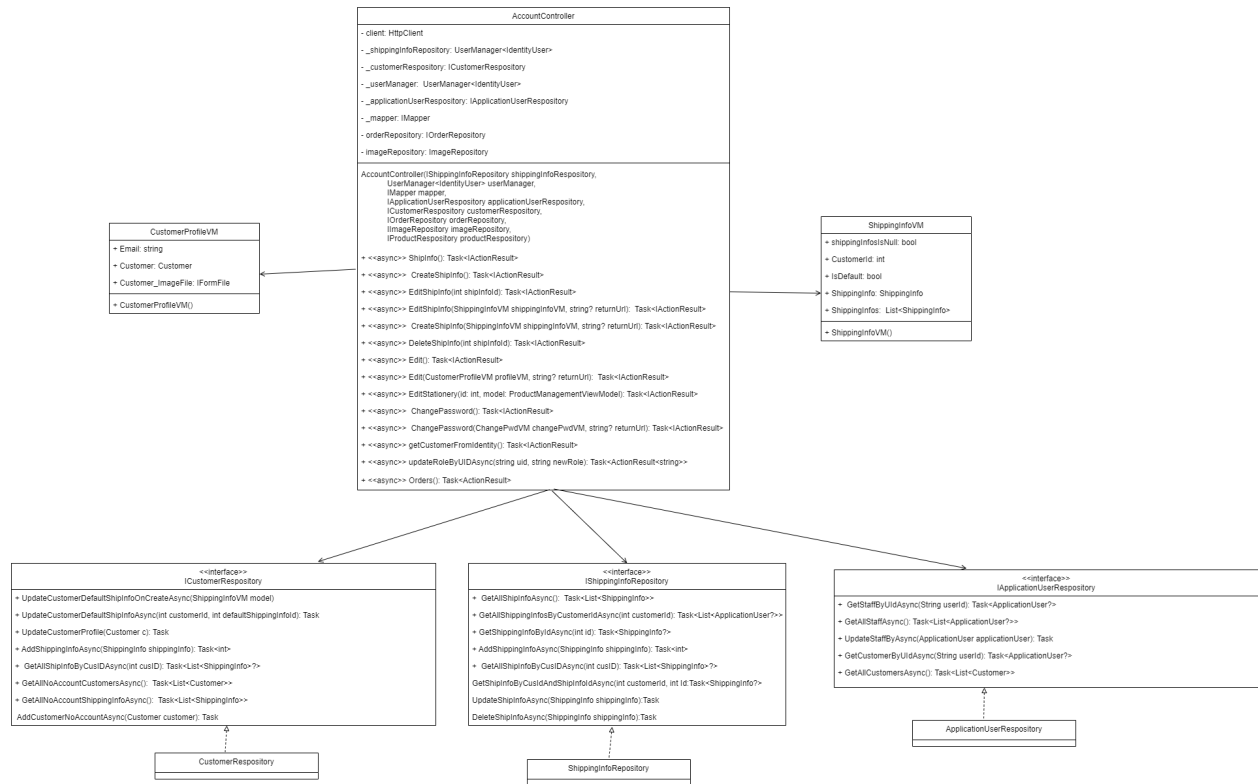# 3.11 (Customer)Shipping Information Management

3.11.1 Class Diagram

Figure 3.11.1.1: Create new shipping information functions class diagram

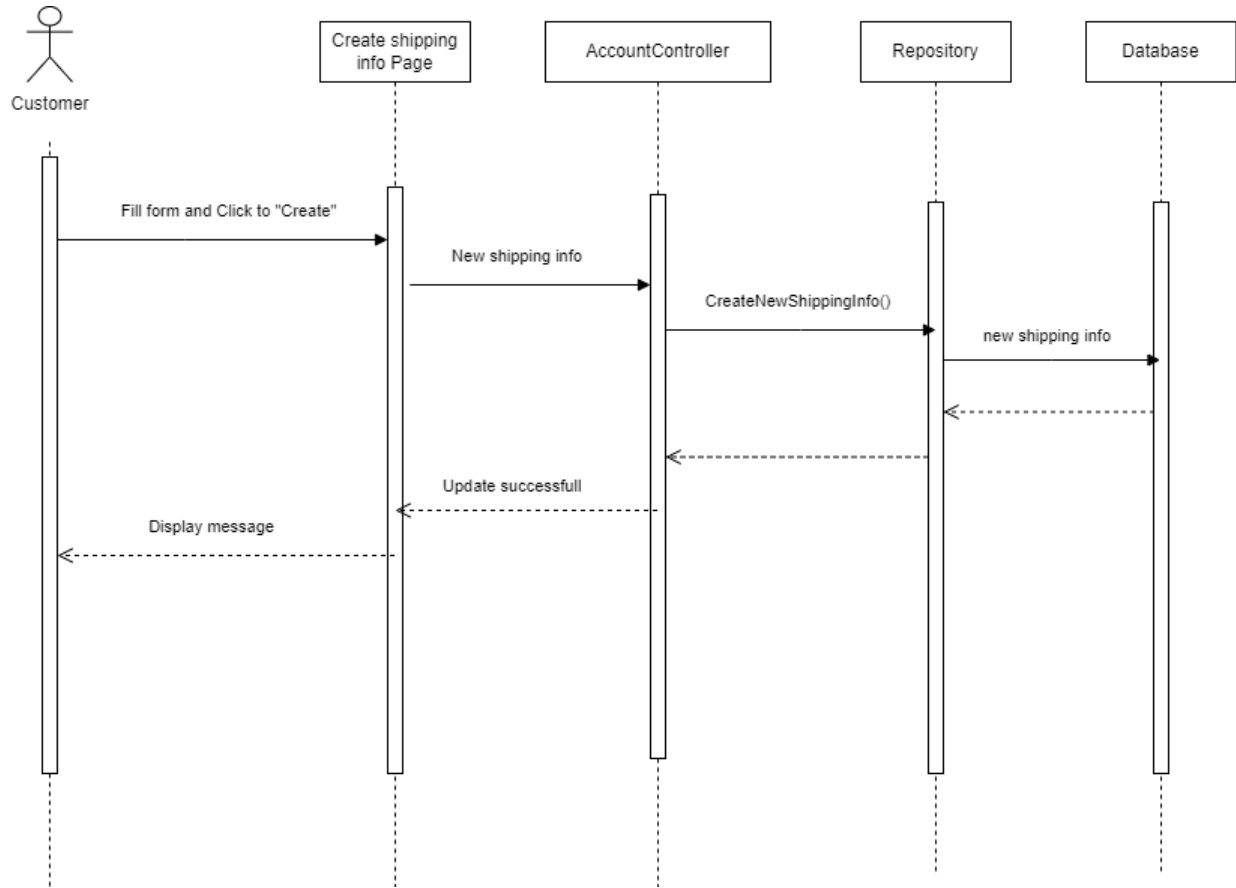### 3.11.2 Create new shipping information

Figure 3.11.2.1: Create new shipping information sequence diagram

3.11.3 Edit shipping information
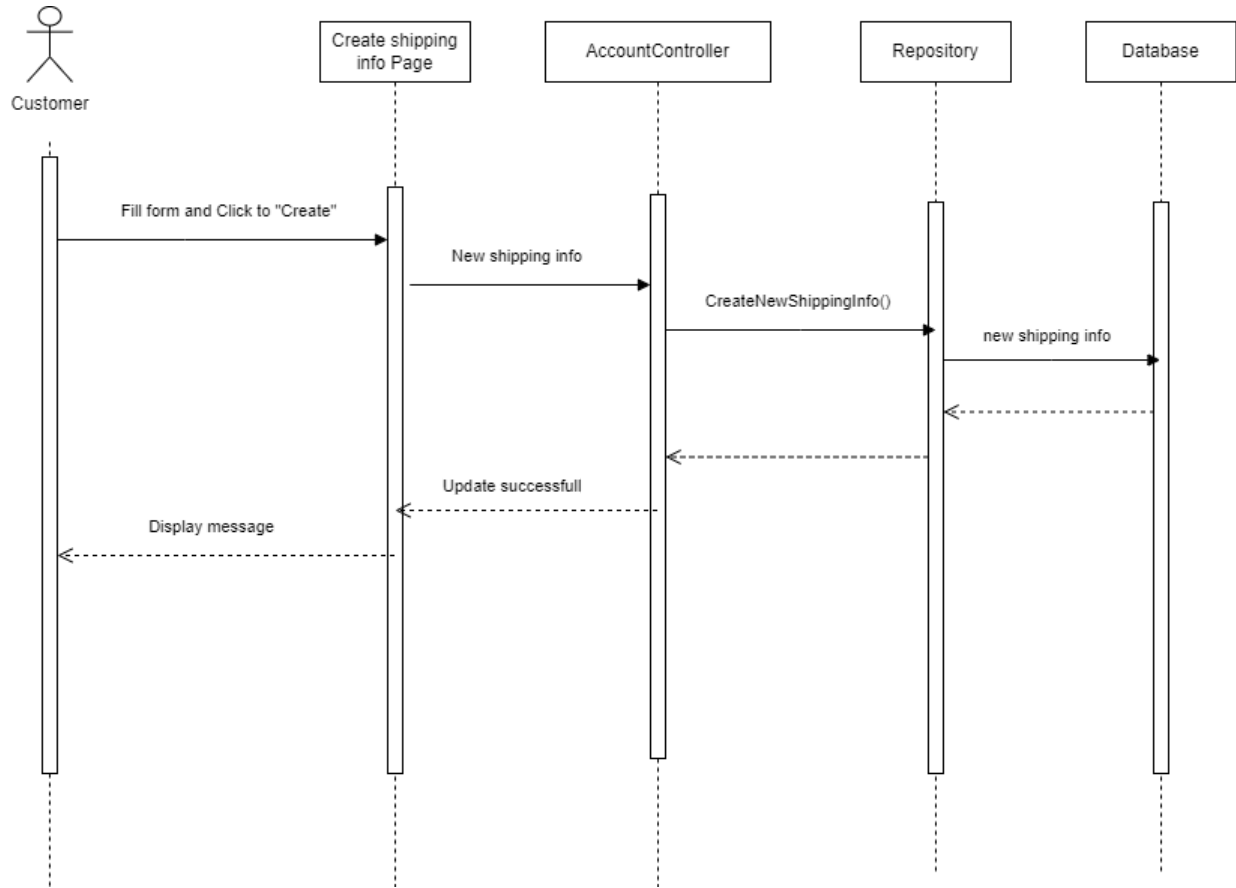
Figure 3.11.3.1: Edit shipping information sequence diagram

3.11.4 Delete shipping information
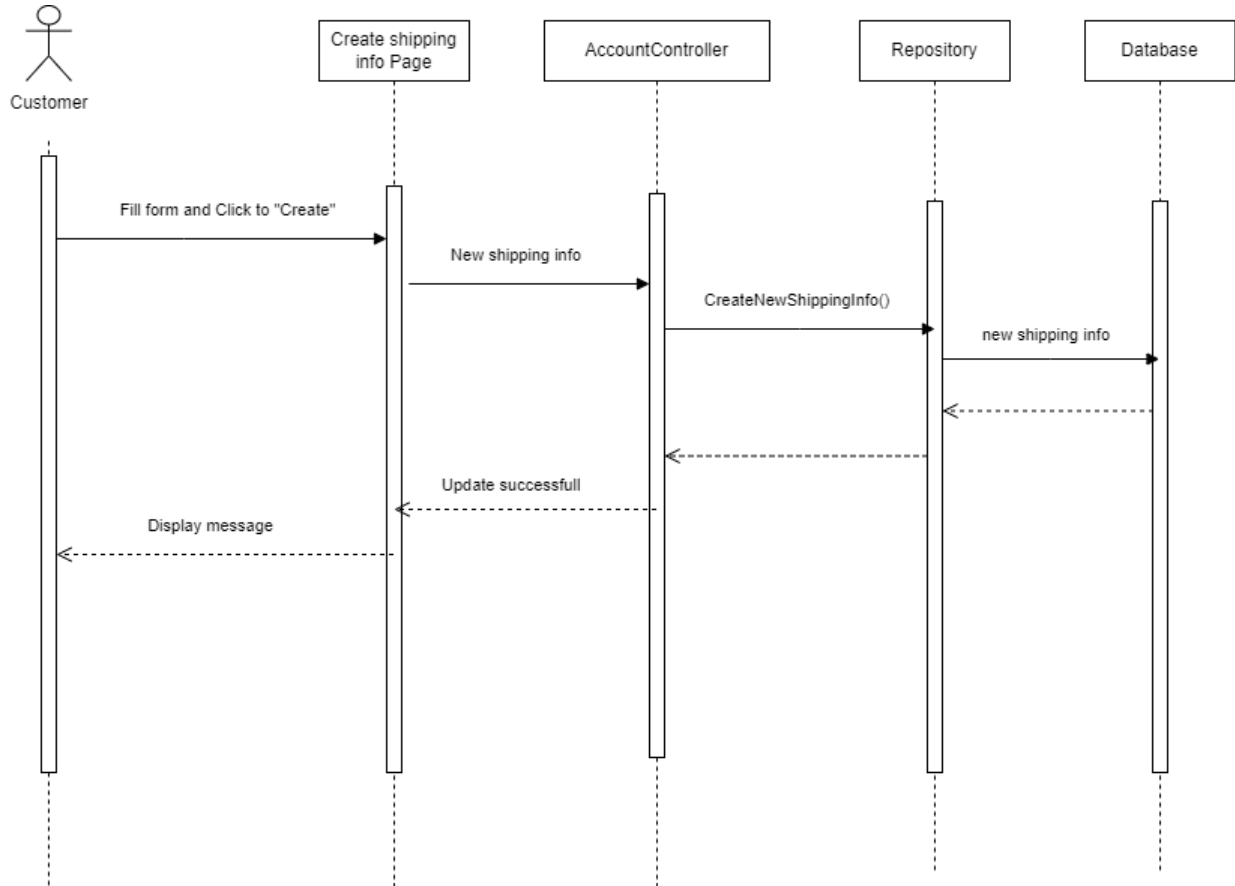
Figure 3.11.4.1: Delete shipping information sequence diagram

3.12 Supplier Management

3.12.1 Class Diagram
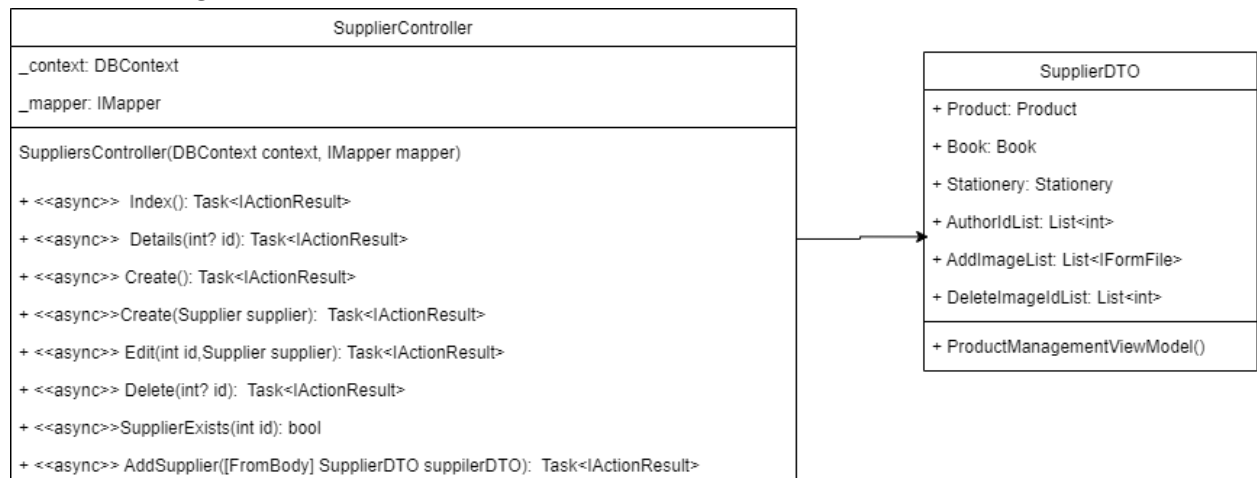
Figure 3.12.1.1: Supplier Management functions class diagram

3.12.2 View list of supplier
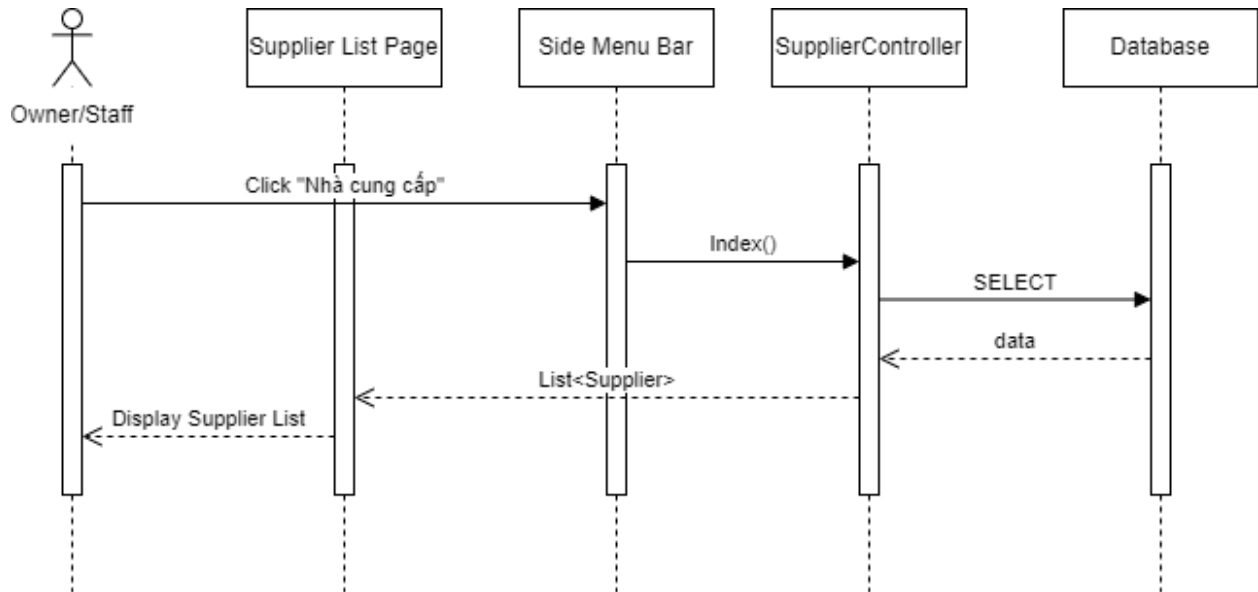
Figure 3.12.2.1: View list of supplier sequence diagram

3.12.3 Update supplier

Figure 3.12.3.1: Update supplier sequence diagram

3.12.4 View details supplier

Figure 3.12.4.1: View details supplier sequence diagram
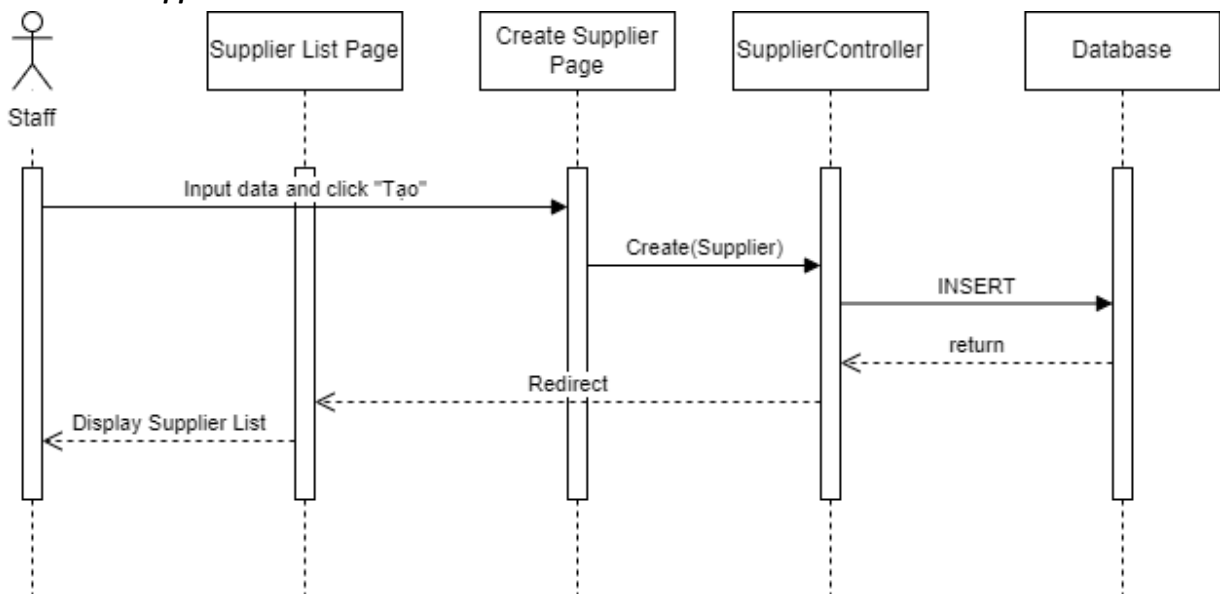
3.12.5 Create supplier

Figure 3.12.5.1: Create supplier sequence diagram
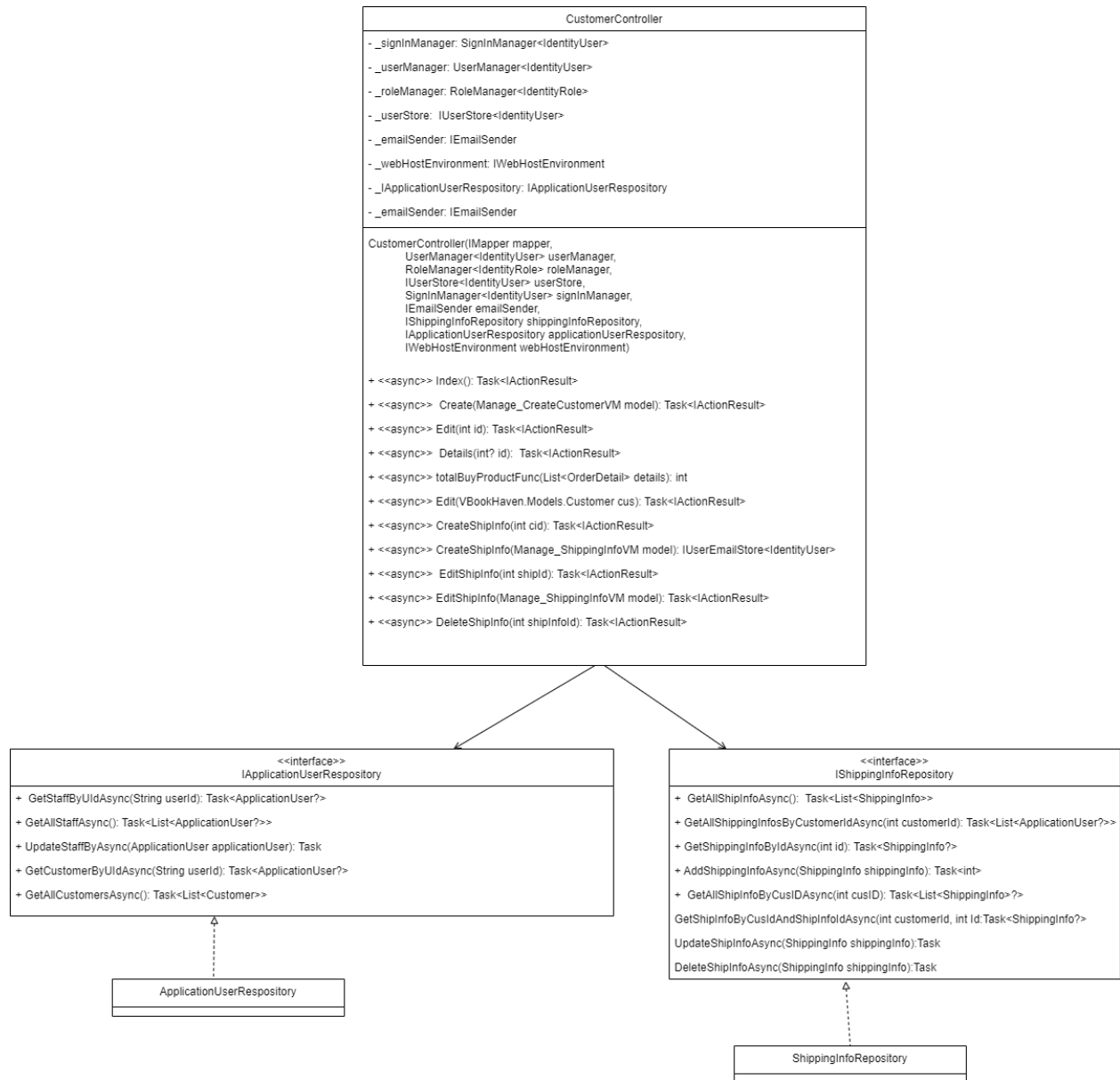
3.13 Customer Management

3.13.1 Class Diagram

CustomerController

- _signInManager: SignInManager<IdentityUser>
- _userManager: UserManager<IdentityUser>
- _roleManager: RoleManager<IdentityRole>
- _userStore: IUserStore<IdentityUser>
- _emailSender: IEmailSender
- _webHostEnvironment: IWebHostEnvironment
- _IApplicationUserRespository: IApplicationUserRespository
- _emailSender: IEmailSender

CustomerController(IMapper mapper,
        UserManager<IdentityUser> userManager,
        RoleManager<IdentityRole> roleManager,
        IUserStore<IdentityUser> userStore,
        SignInManager<IdentityUser> signInManager,
        IEmailSender emailSender,
        IShippingInfoRepository shippingInfoRepository,
        IApplicationUserRespository applicationUserRespository,
        IWebHostEnvironment webHostEnvironment)

+ <<async>> Index(): Task<IActionResult>
+ <<async>> Create(Manage_CreateCustomerVM model): Task<IActionResult>
+ <<async>> Edit(int id): Task<IActionResult>
+ <<async>> Details(int? id): Task<IActionResult>
+ <<async>> totalBuyProductFunc(List<OrderDetail> details): int
+ <<async>> Edit(VBookHaven.Models.Customer cus): Task<IActionResult>
+ <<async>> CreateShipInfo(int cid): Task<IActionResult>
+ <<async>> CreateShipInfo(Manage_ShippingInfoVM model): IUserEmailStore<IdentityUser>
+ <<async>> EditShipInfo(int shipId): Task<IActionResult>
+ <<async>> EditShipInfo(Manage_ShippingInfoVM model): Task<IActionResult>
+ <<async>> DeleteShipInfo(int shipInfoId): Task<IActionResult>

<<interface>>
IApplicationUserRespository

+ GetStaffByUIdAsync(String userId): Task<ApplicationUser?>
+ GetAllStaffAsync(): Task<List<ApplicationUser?>>
+ UpdateStaffByAsync(ApplicationUser applicationUser): Task
+ GetCustomerByUIdAsync(String userId): Task<ApplicationUser?>
+ GetAllCustomersAsync(): Task<List<Customer>>

ApplicationUserRespository

<<interface>>
IShippingInfoRepository

+ GetAllShipInfoAsync(): Task<List<ShippingInfo>>
+ GetAllShippingInfosByCustomerIdAsync(int customerId): Task<List<ApplicationUser?>>
+ GetShippingInfoByIdAsync(int id): Task<ShippingInfo?>
+ AddShippingInfoAsync(ShippingInfo shippingInfo): Task<int>
+ GetAllShipInfoByCusIDAsync(int cusID): Task<List<ShippingInfo>?>
GetShipInfoByCusIdAndShipInfoIdAsync(int customerId, int Id:Task<ShippingInfo?>
UpdateShipInfoAsync(ShippingInfo shippingInfo):Task
DeleteShipInfoAsync(ShippingInfo shippingInfo):Task

ShippingInfoRepository

Figure 3.13.1.1: Dashboard functions class diagram
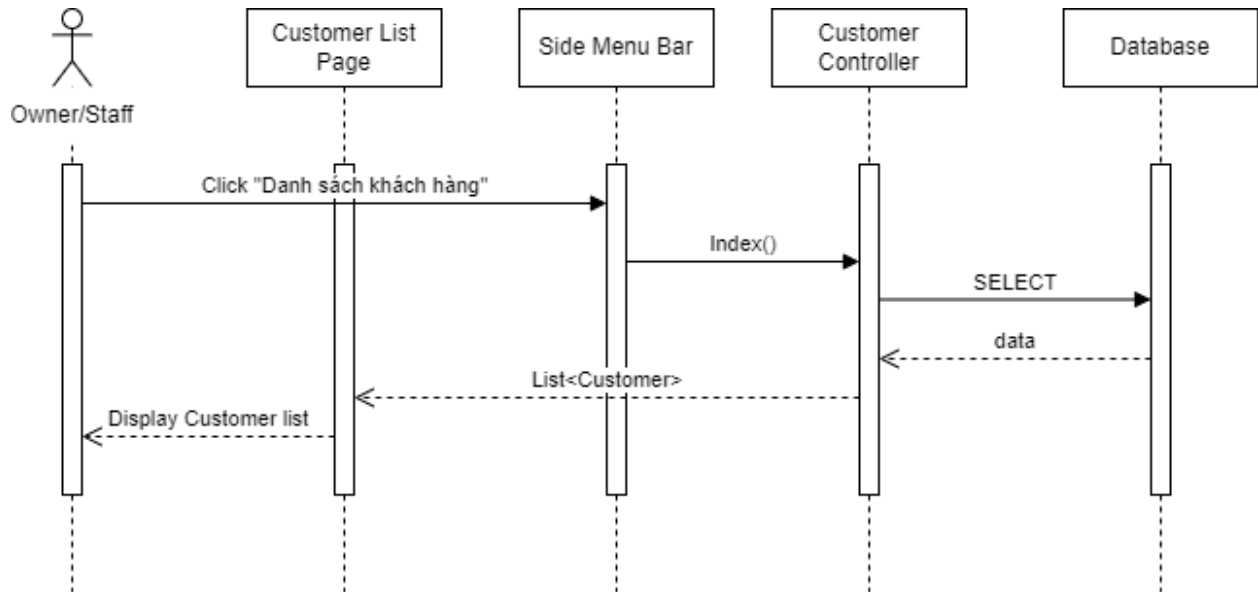
3.13.2 View list of customer

Figure 3.13.2.1: View list of customer sequence diagram
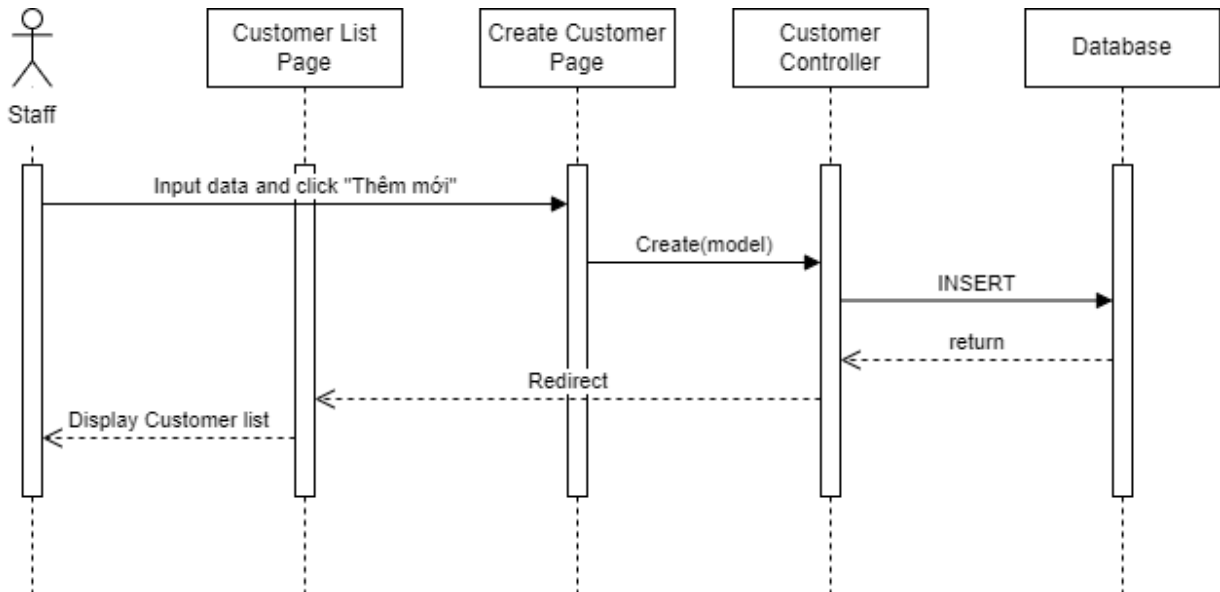
3.13.3 Create a customer

Figure 3.13.3.1: Create a customer sequence diagram
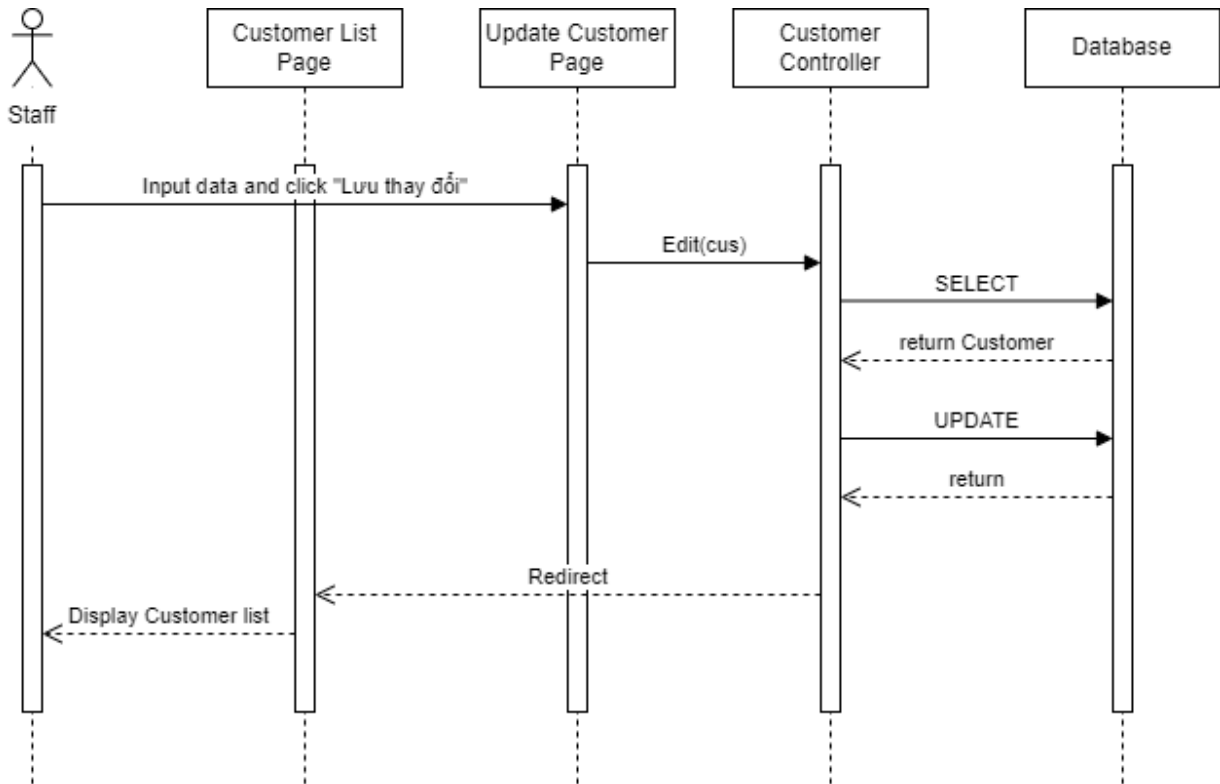
3.13.4 Update a customer

Figure 3.13.4.1: Update a customer sequence diagram
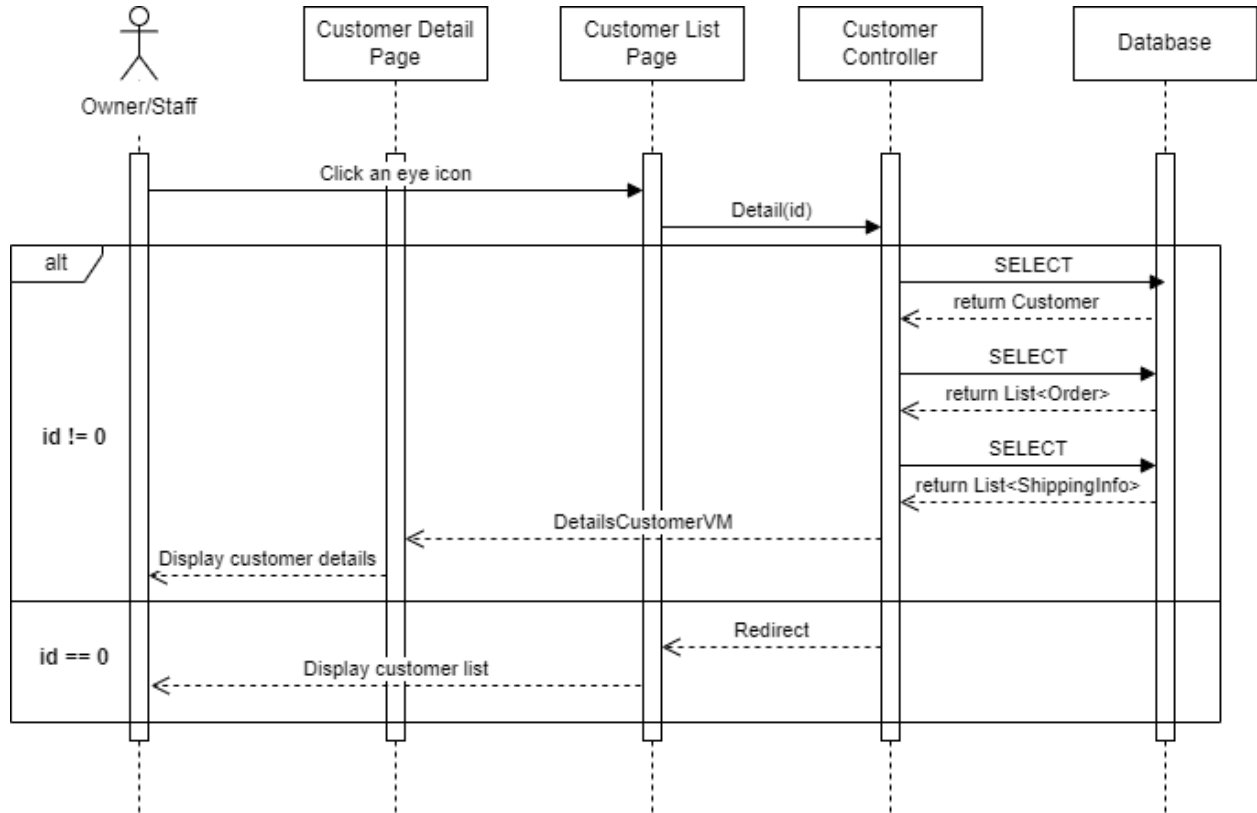
3.13.5 View details of customer

Figure 3.13.5.1: View details of customer sequence diagram

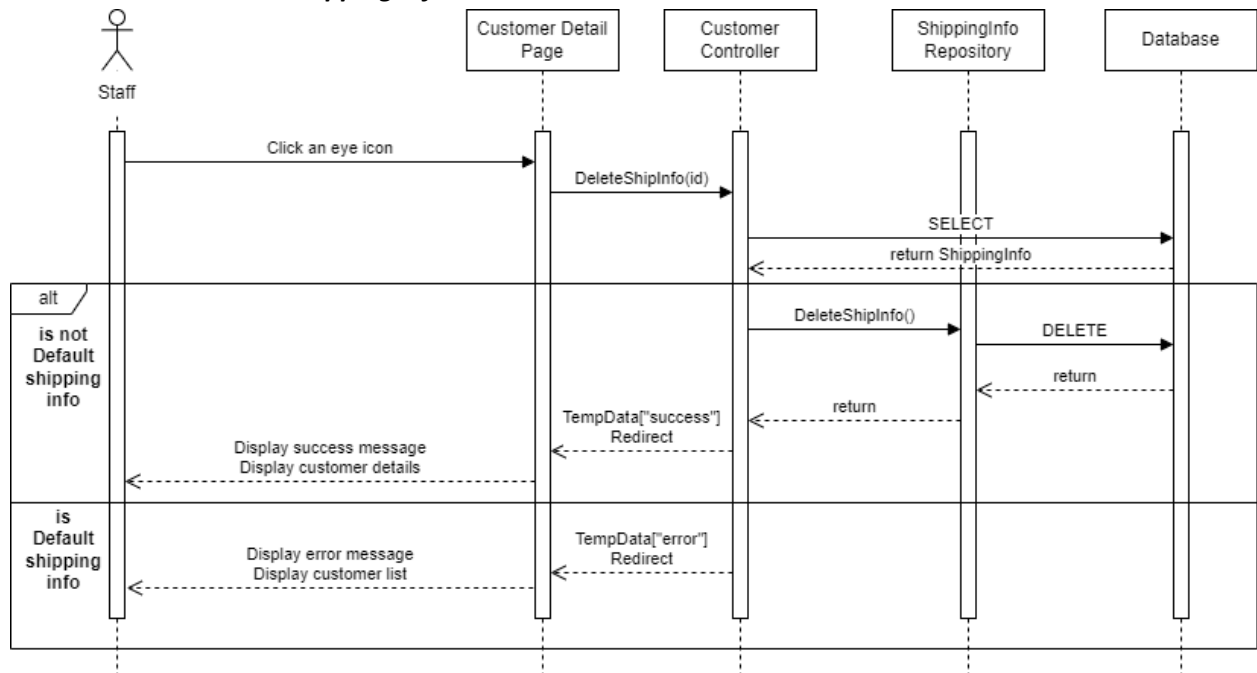### 3.13.6 Delete customer shipping information

Figure 3.13.6.1: Delete customer shipping information sequence diagram

3.13.7 Create customer shipping information
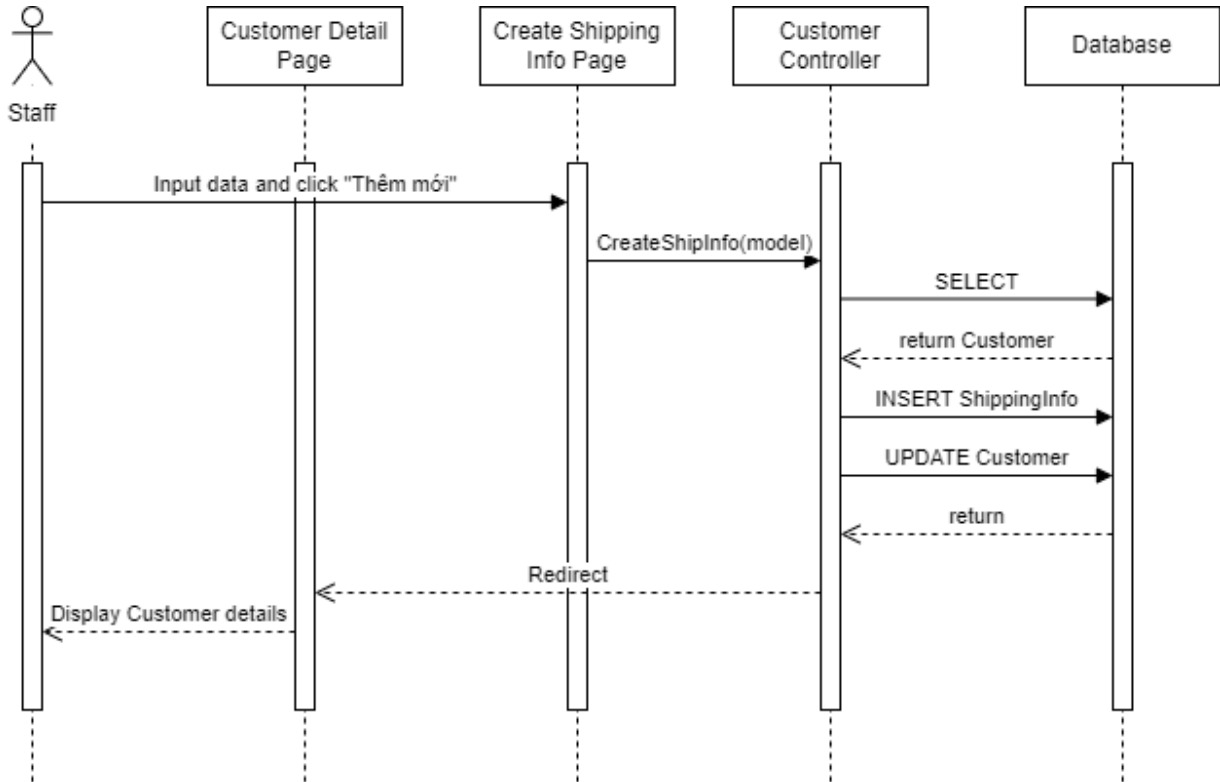
Figure 3.13.7.1: Create customer shipping information sequence diagram

3.13.8 Edit customer shipping information

3.14. Dashboard

3.14.1 Class Diagram
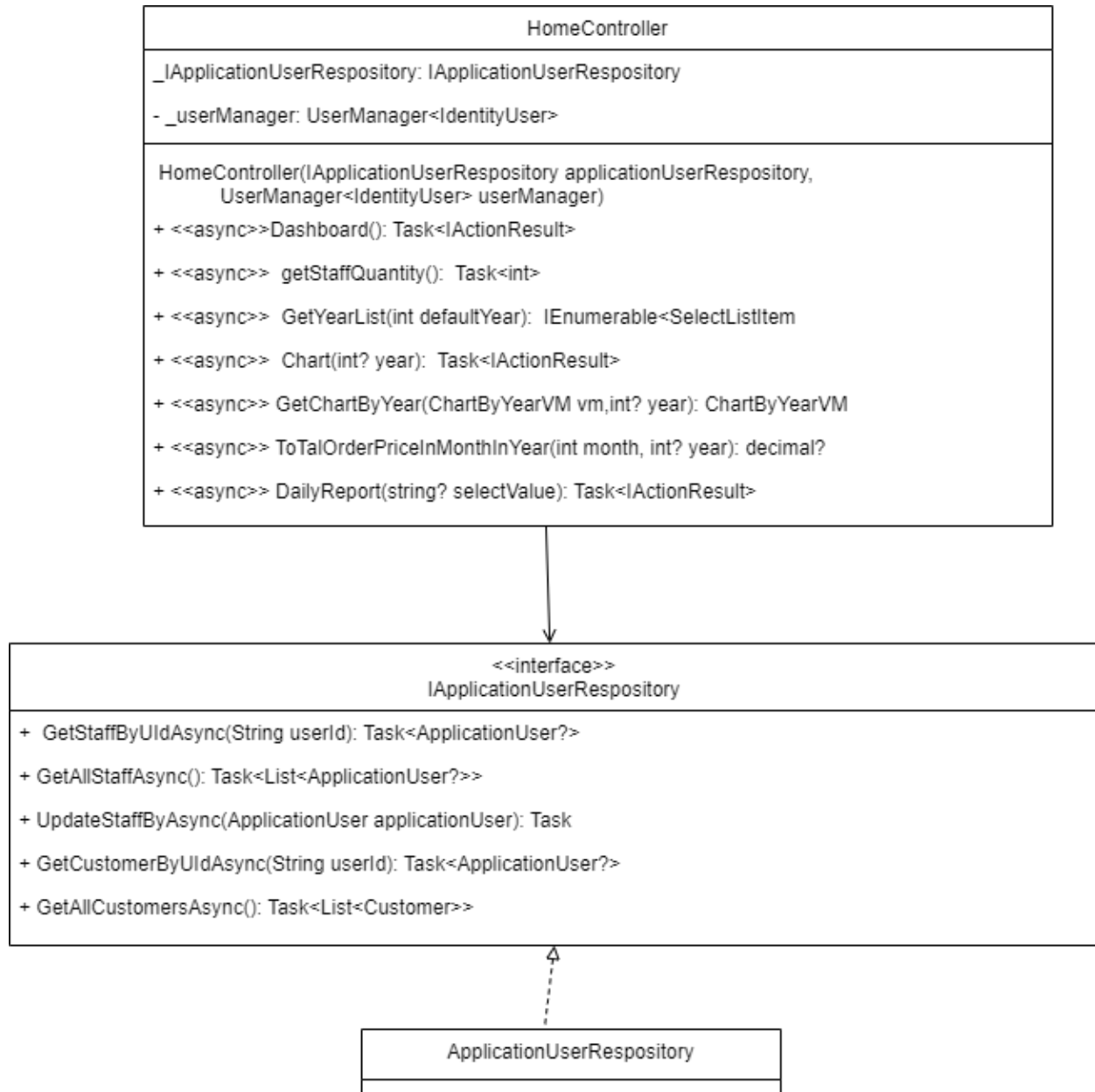
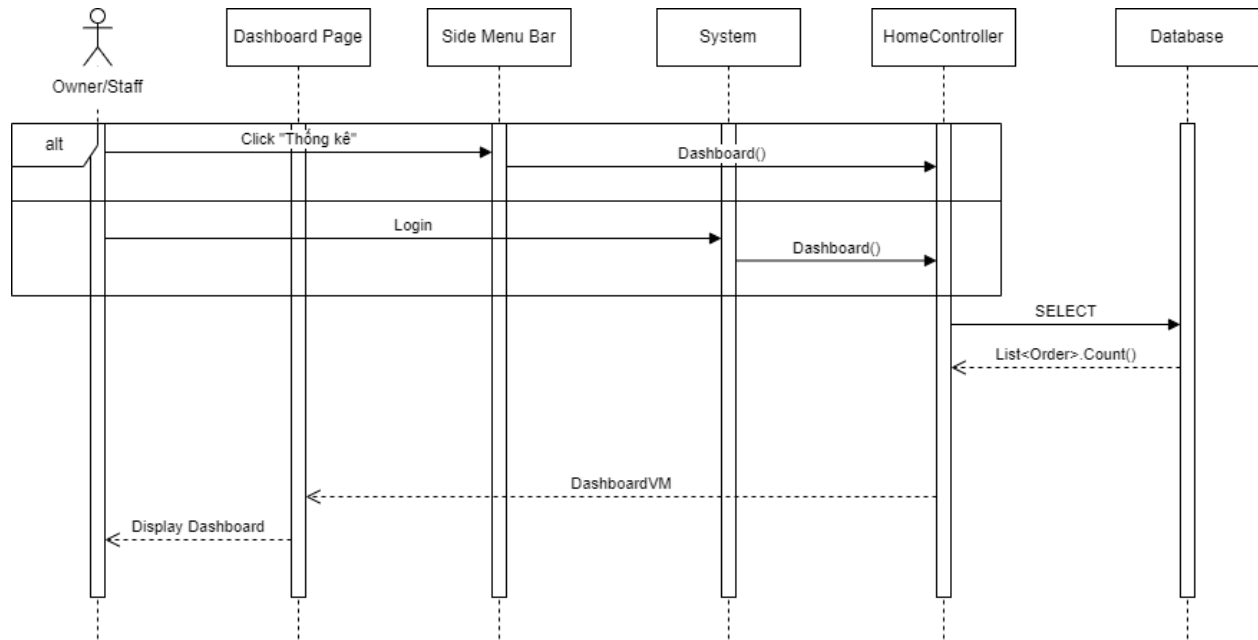*Figure 3.14.1.1: Dashboard functions class diagram*

3.14.2 View dashboard

Figure 3.14.2.1: View dashboard sequence diagram