Assignment 3

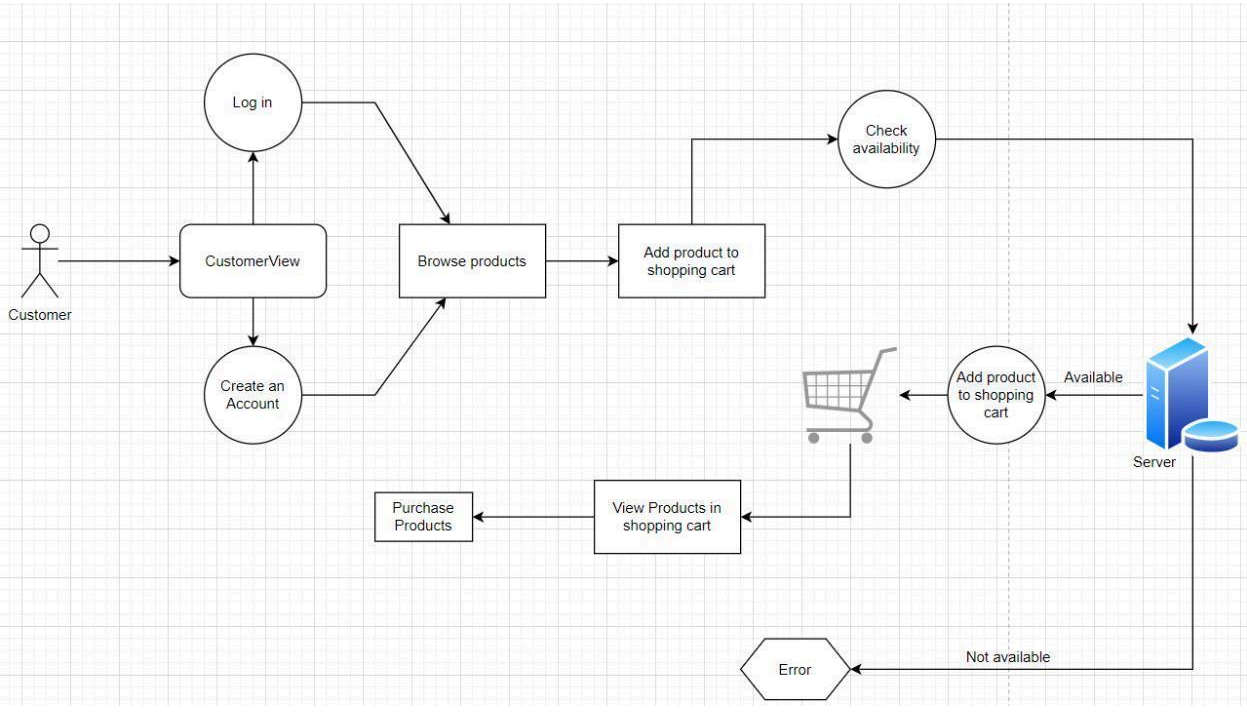
Description

Similar to A2, except that I implemented all the new design details such as factory pattern, authorization pattern, front controller pattern, etc. The use cases haven't changed, and the actual use of the program is very similar to A2 with some improvements to the user experience. However, the domain models and the class diagrams are different and they show the new design of the application.

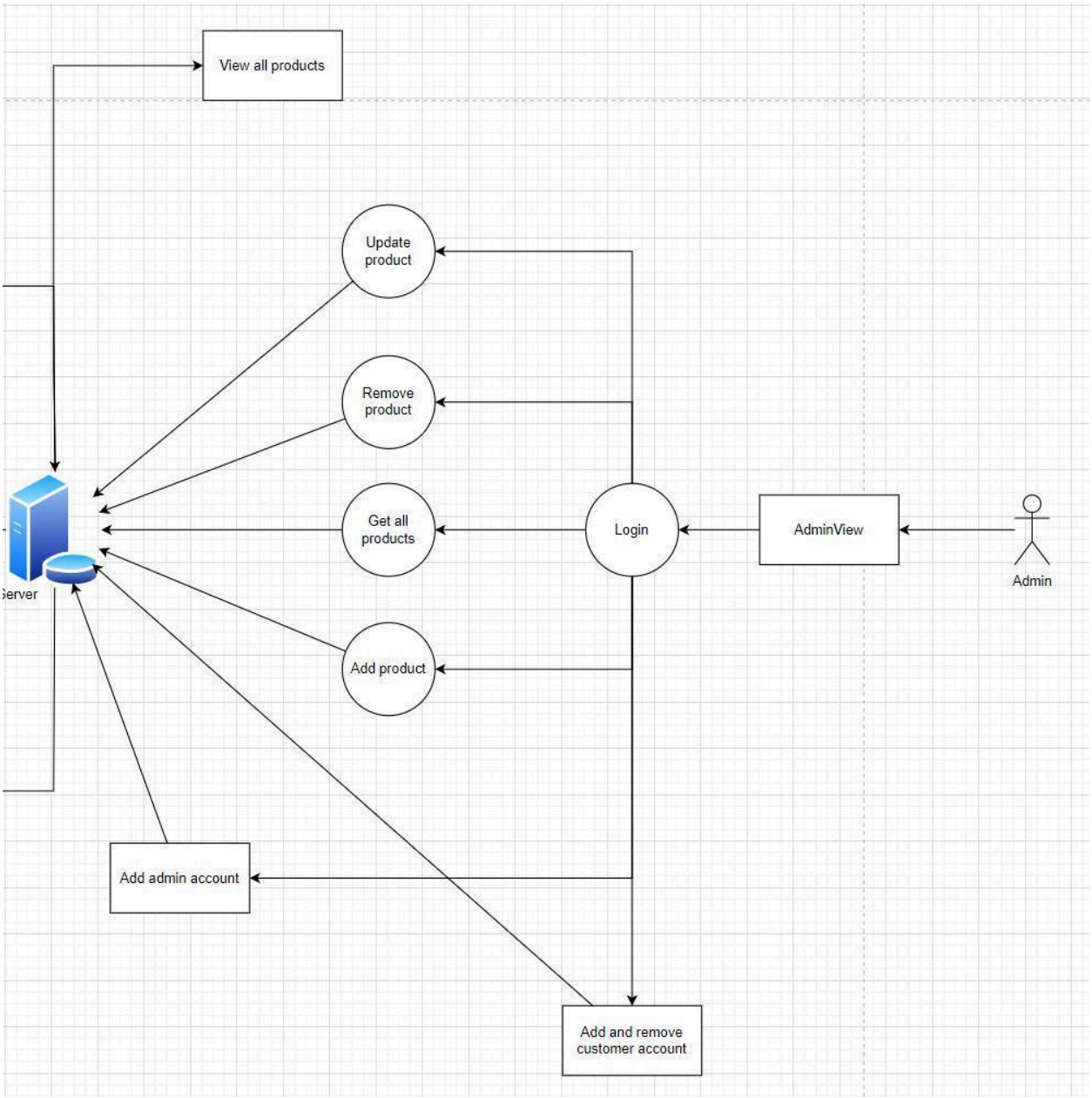
Use Cases

The use case shown below demonstrates the interaction between customers and users with the system. It shows what happens when the customer makes a request to the server, when the admin makes a request to the server, and the differences between how the server responds in both scenarios.

Customer use case:

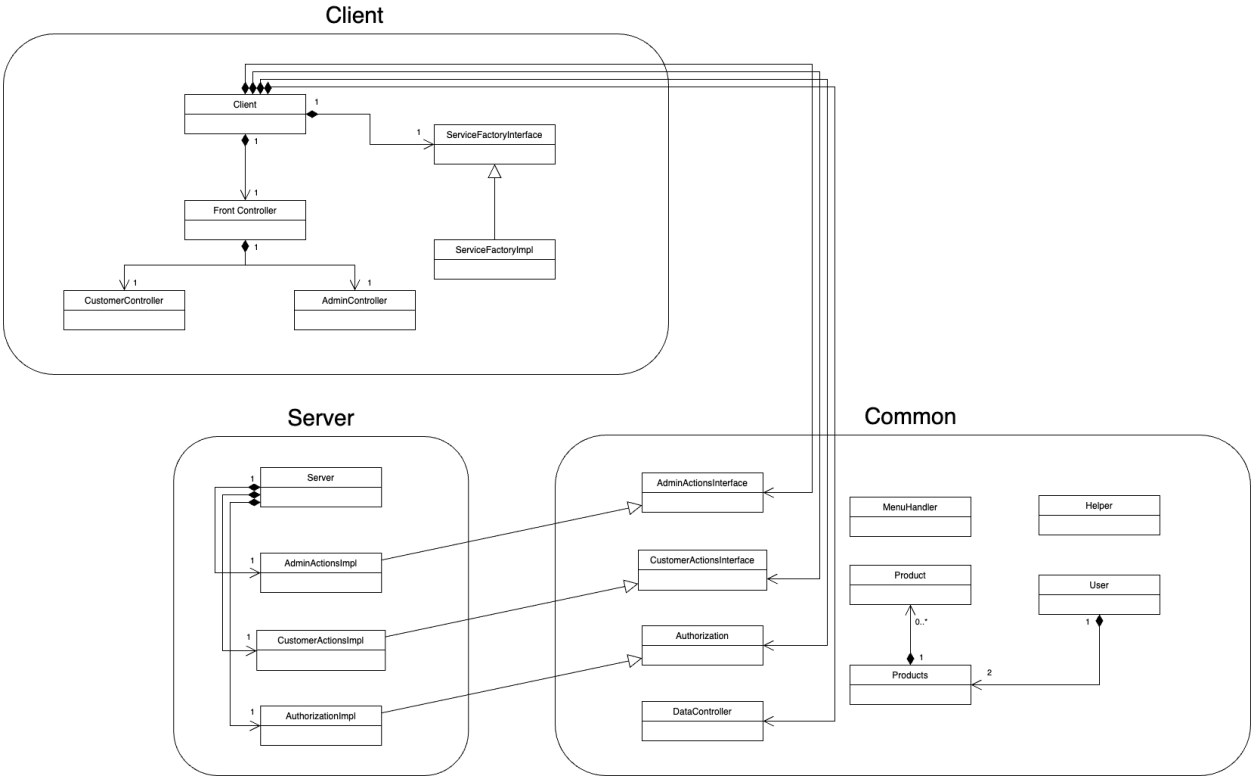


Admin use case:



Domain Model

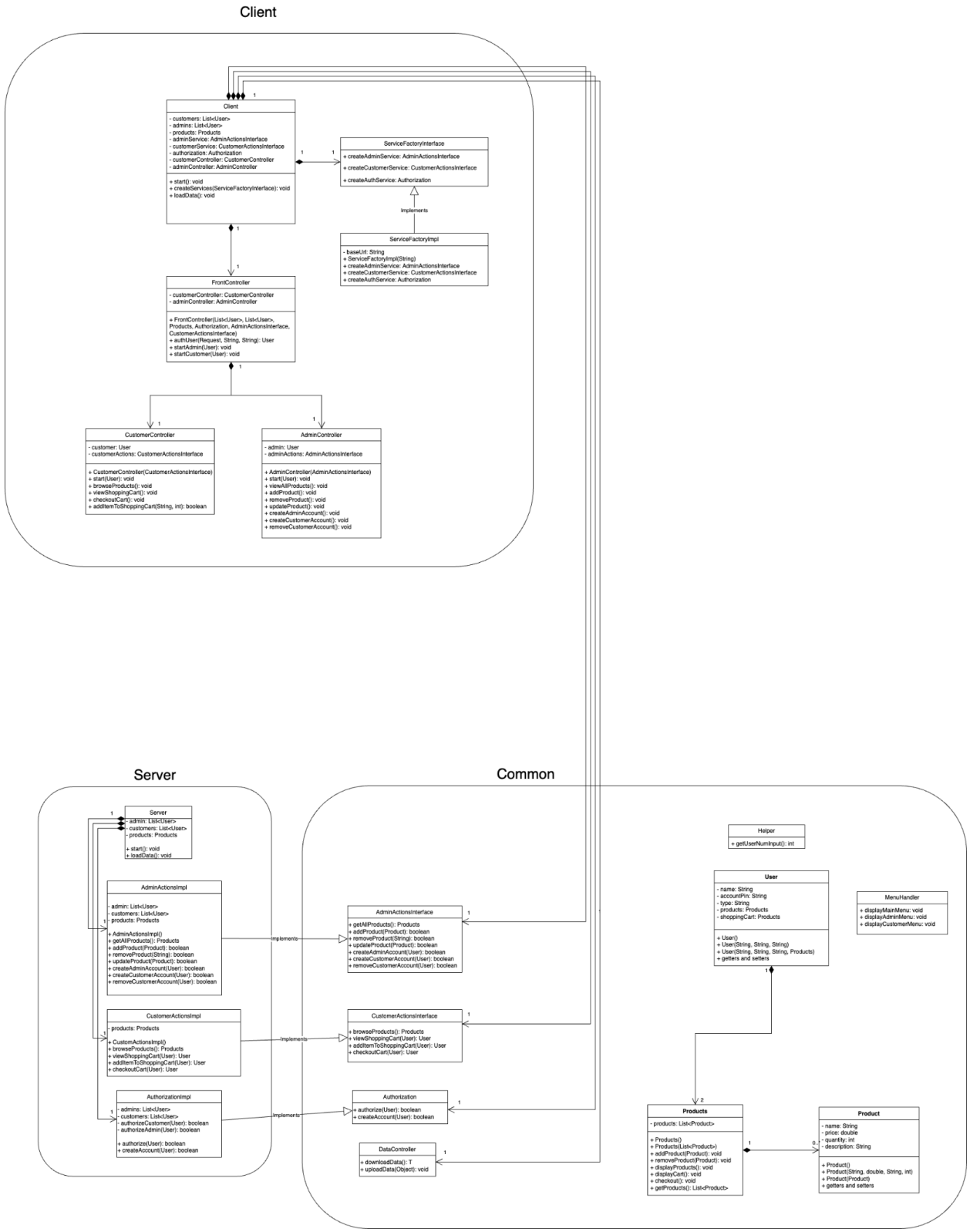
The domain model is a different version of class diagrams as it does not show the very details of the system, but rather, it shows how the different component overall interact with each other. The domain model is an easy and a quick way for someone to understand the overall architecture of the system. A domain model diagram visualizes the conceptual model of a system, focusing on the domain and its entities and their relationships. I didn't need to divide them into client, common, and server, but I thought it would be easier to show that way.



Class Diagram

A class diagram, part of UML, represents the structure of a system by showing its classes, attributes, operations, and the relationships between classes. Class diagrams implement a demo of an online store, where there are two types of user interaction. The first being admin, and the second is customer. Naturally, the admin interaction with the software is different from the customer interaction with the software, they have different authorization levels and different data that they control. For example, a user has a shopping cart that they can put products into, and they also purchase products from the shopping cart, while an admin cannot do the same. The class diagram shown below presents the interactions between the different classes with each other in the system.

The picture doesn't show the details, but a copy of the .drawio file has been attached to the submission zip as well. Also, I didn't need to divide them into client, common, and server, but I thought it would be easier to show that way.


Client
Client
- customers: List<User>
- admins: List<User>
- products: Products
- adminService: AdminActionsInterface
- customerService: CustomerActionsInterface
- authorization: Authorization
- customerController: CustomerController
- adminController: AdminController
+ start(): void
+ createServices(ServiceFactoryInterface): void
+ loadData(): void
ServiceFactoryInterface
+ createAdminService: AdminActionsInterface
+ createCustomerService: CustomerActionsInterface
+ createAuthService: Authorization
Implements
ServiceFactoryImpl
- baseUrl: String
+ ServiceFactoryImpl(String)
+ createAdminService: AdminActionsInterface
+ createCustomerService: CustomerActionsInterface
+ createAuthService: Authorization
FrontController
- customerController: CustomerController
- adminController: AdminController
+ FrontController(List<User>, List<User>, Products, Authorization, AdminActionsInterface, CustomerActionsInterface)
+ authUser(Request, String, String): User
+ startAdmin(User): void
+ startCustomer(User): void
CustomerController
- customer: User
- customerActions: CustomerActionsInterface
+ CustomerController(CustomerActionsInterface)
+ start(User): void
+ browseProducts(): void
+ viewShoppingCart(): void
+ checkoutCart(): void
+ addItemToShoppingCart(String, int): boolean
AdminController
- admin: User
- adminActions: AdminActionsInterface
+ AdminController(AdminActionsInterface)
+ start(User): void
+ viewAllProducts(): void
+ addProduct(): void
+ removeProduct(): void
+ updateProduct(): void
+ createAdminAccount(): void
+ createCustomerAccount(): void
+ removeCustomerAccount(): void
Server
Server
- admin: List<User>
- customers: List<User>
- products: Products
+ start(): void
+ loadData(): void
AdminActionsImpl
- admin: List<User>
- customers: List<User>
- products: Products
+ AdminActionsImpl()
+ getAllProducts(): Products
+ addProduct(Product): boolean
+ removeProduct(String): boolean
+ updateProduct(Product): boolean
+ createAdminAccount(User): boolean
+ createCustomerAccount(User): boolean
+ removeCustomerAccount(User): boolean
CustomerActionsImpl
- products: Products
+ CustomActionsImpl()
+ browseProducts(): Products
+ viewShoppingCart(User): User
+ addItemToShoppingCart(User): User
+ checkoutCart(User): User
AuthorizationImpl
- admins: List<User>
- customers: List<User>
- authorizeCustomer(User): boolean
- authorizeAdmin(User): boolean
+ authorize(User): boolean
+ createAccount(User): boolean
Implements
Implements
Implements
Common
AdminActionsInterface
+ getAllProducts(): Products
+ addProduct(Product): boolean
+ removeProduct(String): boolean
+ updateProduct(Product): boolean
+ createAdminAccount(User): boolean
+ createCustomerAccount(User): boolean
+ removeCustomerAccount(User): boolean
CustomerActionsInterface
+ browseProducts(): Products
+ viewShoppingCart(User): User
+ addItemToShoppingCart(User): User
+ checkoutCart(User): User
Authorization
+ authorize(User): boolean
+ createAccount(User): boolean
DataController
+ downloadData(): T
+ uploadData(Object): void
Helper
+ getUserNumInput(): int
User
- name: String
- accountPin: String
- type: String
- products: Products
- shoppingCart: Products
+ User()
+ User(String, String, String)
+ User(String, String, String, Products)
+ getters and setters
MenuHandler
+ displayMainMenu: void
+ displayAdminMenu: void
+ displayCustomerMenu: void
Products
- products: List<Product>
+ Products()
+ Products(List<Product>)
+ addProduct(Product): void
+ removeProduct(Product): void
+ displayProducts(): void
+ displayCart(): void
+ checkout(): void
+ getProducts(): List<Product>
Product
- name: String
- price: double
- quantity: int
- description: String
+ Product()
+ Product(String, double, String, int)
+ Product(Product)
+ getters and setters

Program Running

```
oabdela@in-csci-rrpc02:~/A3$ make server
javac common/*.java
javac client/*.java
javac server/*.java
java server/Server
Server is running...
Got request to authorize user: admin
Admin authorized: admin
Received request to get all products
```

```
oabdela@in-csci-rrpc01:~/A3$ make client
javac common/*.java
javac client/*.java
javac server/*.java
java client/Client
---------------------------------Welcome to our online store---------------------------------

1- Sign in as admin
2- Sign in as user
3- Create an account
4- Exit

Please enter a number:
1
Enter username: admin
Enter pin: 12345


1. View products
2. Add a product
3. Remove a product
4. Update a product
5. Create an admin account
6. Create a customer account
7. Remove a customer account
8. Exit

Enter your choice: 1
name: Apple
price: 1.0
description: Delicious
Quantity: 500


1. View products
2. Add a product
3. Remove a product
4. Update a product
5. Create an admin account
6. Create a customer account
7. Remove a customer account
8. Exit
```