# CSCI 5448: Object-Oriented Analysis & Design

## Project 7: E-Commerce Web Application

**Anuj Bhavani Shankar | Rajiv Srivatsava Vangala**

### Final State of System Statement

After Project 7, our application allows the creation of a User(Customer/Vendor) and allows the user to perform specific tasks based on their role. Here is the list of requirements defined by us as part of Project 5:

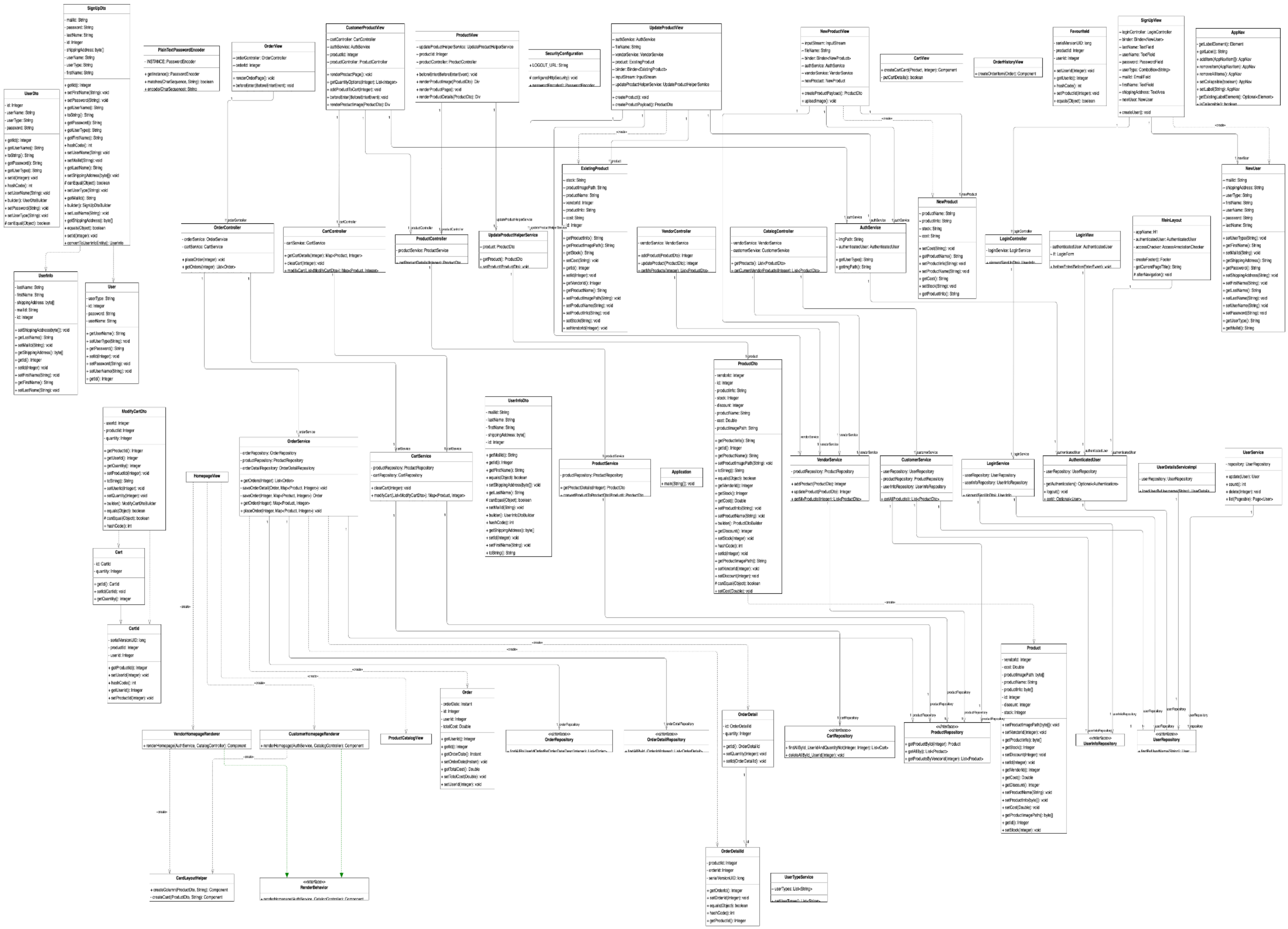
1. Users should be able to Sign up as a customer or a vendor.
2. An existing user should be able to login.
3. A customer should be able to view a catalog of products.
4. A customer should be able to filter items based on vendor.
5. A customer should be able to mark any product as favorite.
6. A customer should be able to place an order with one or more item(s) in their cart.
7. A customer should be able to view their order history.
8. A vendor should be able to view and modify their inventory.
9. A vendor should be able to add to their inventory.

Out of the above requirements, we implemented 7 of them excluding points 4 and 5.

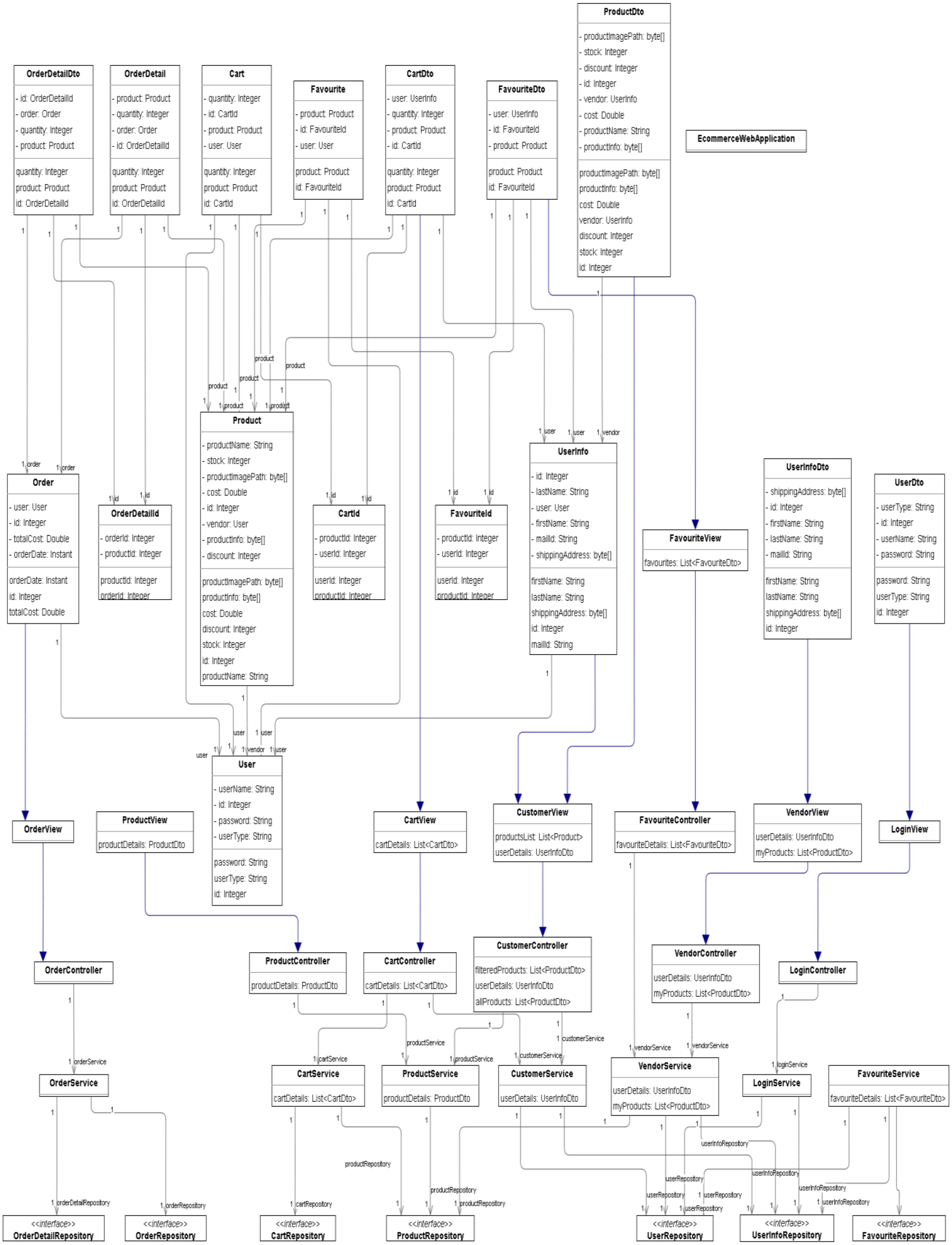
The favorites feature could not be completed as it involved extensive UI work and hence had to be scoped out. The filter items feature was partially done, that is, we had the API in place, but we ran out of time for implementing the UI portion.

### Final Class Diagram and Comparison Statement

The following is the final UML Class Diagram after our implementation for Project 7.
We have also attached the same image in the following Drive link:
https://drive.google.com/file/d/15xxviv6hoKK-Wzzl7wii_RnoAsw4w0kM/view?usp=share_link


SignUpDto
- mailId: String
- password: String
- lastName: String
- id: Integer
- shippingAddress: byte[]
- userName: String
- userType: String
- firstName: String
+ getId(): Integer
+ setFirstName(String): void
+ setPassword(String): void
+ getUserName(): String
+ toString(): String
+ getPassword(): String
+ getUserType(): String
+ getFirstName(): String
+ hashCode(): int
+ setUserName(String): void
+ setMailId(String): void
+ getLastName(): String
+ setShippingAddress(byte[]): void
# canEqual(Object): boolean
+ setUserType(String): void
+ getMailId(): String
+ builder(): SignUpDtoBuilder
+ setLastName(String): void
+ getShippingAddress(): byte[]
+ equals(Object): boolean
+ setId(Integer): void
+ convertToUserInfoEntity(): UserInfo
UserDto
- id: Integer
- userName: String
- userType: String
- password: String
+ getId(): Integer
+ getUserName(): String
+ toString(): String
+ getPassword(): String
+ getUserType(): String
+ setId(Integer): void
+ hashCode(): int
+ setUserName(String): void
+ builder(): UserDtoBuilder
+ setPassword(String): void
+ setUserType(String): void
# canEqual(Object): boolean
UserInfo
- lastName: String
- firstName: String
- shippingAddress: byte[]
- mailId: String
- id: Integer
+ setShipingAddress(byte[]): void
+ getLastName(): String
+ setMailId(String): void
+ getShippingAddress(): byte[]
+ getId(): Integer
+ setId(Integer): void
+ setFirstName(String): void
+ getFirstName(): String
+ setLastName(String): void
User
- userType: String
- id: Integer
- password: String
- userName: String
+ getUserName(): String
+ setUserType(String): void
+ getPassword(): String
+ setId(Integer): void
+ setPassword(String): void
+ setUserName(String): void
+ getId(): Integer
PlainTextPasswordEncoder
- INSTANCE: PasswordEncoder
+ getInstance(): PasswordEncoder
+ matches(CharSequence, String): boolean
+ encode(CharSequence): String
OrderView
~ orderController: OrderController
~ orderId: Integer
+ renderOrderPage(): void
+ beforeEnter(BeforeEnterEvent): void
CustomerProductView
~ cartController: CartController
~ authService: AuthService
~ productId: Integer
~ productController: ProductController
+ renderProductPage(): void
+ getQuantityOptions(Integer): List<Integer>
+ addProductToCart(Integer): void
+ beforeEnter(BeforeEnterEvent): void
+ renderProductImage(ProductDto): Div
ProductView
~ updateProductHelperService: UpdateProductHelperService
~ productId: Integer
~ productController: ProductController
+ beforeEnter(BeforeEnterEvent): void
+ renderProductImage(ProductDto): Div
+ renderProductPage(): void
+ renderProductDetails(ProductDto): Div
SecurityConfiguration
+ LOGOUT_URL: String
# configure(HttpSecurity): void
+ passwordEncoder(): PasswordEncoder
UpdateProductView
~ authService: AuthService
~ fileName: String
~ vendorService: VendorService
~ product: ExistingProduct
~ binder: Binder<ExistingProduct>
~ inputStream: InputStream
~ updateProductHelperService: UpdateProductHelperService
+ createProduct(): void
+ createProductPayload(): ProductDto
NewProductView
~ inputStream: InputStream
~ fileName: String
~ binder: Binder<NewProduct>
~ authService: AuthService
~ vendorService: VendorService
~ newProduct: NewProduct
+ createProductPayload(): ProductDto
+ uploadImage(): void
CartView
+ createCartCard(Product, Integer): Component
- putCartDetails(): boolean
OrderHistoryView
+ createOrderItem(Order): Component
FavouriteId
- serialVersionUID: long
- productId: Integer
- userId: Integer
+ setUserId(Integer): void
+ getUserId(): Integer
+ hashCode(): int
+ setProductId(Integer): void
+ equals(Object): boolean
SignUpView
~ loginController: LoginController
~ binder: Binder<NewUser>
~ lastName: TextField
~ userName: TextField
~ password: PasswordField
~ userType: ComboBox<String>
~ mailId: EmailField
~ firstName: TextField
~ shippingAddress: TextArea
~ newUser: NewUser
+ createUser(): void
AppNav
- getLabelElement(): Element
+ getLabel(): String
+ addItem(AppNavItem[]): AppNav
+ removeItem(AppNavItem): AppNav
+ removeAllItems(): AppNav
+ setCollapsible(boolean): AppNav
+ setLabel(String): AppNav
- getExistingLabelElement(): Optional<Element>
+ isCollapsible(): boolean
ExistingProduct
~ stock: String
~ productImagePath: String
~ productName: String
~ vendorId: Integer
~ productInfo: String
~ cost: String
~ id: Integer
+ getProductInfo(): String
+ getProductImagePath(): String
+ getStock(): String
+ setCost(String): void
+ getId(): Integer
+ setId(Integer): void
+ getVendorId(): Integer
+ getProductName(): String
+ setProductImagePath(String): void
+ setProductName(String): void
+ setProductInfo(String): void
+ setStock(String): void
+ setVendorId(Integer): void
OrderController
- orderService: OrderService
- cartService: CartService
+ placeOrder(Integer): void
+ getOrders(Integer): List<Order>
CartController
- cartService: CartService
+ getCartDetails(Integer): Map<Product, Integer>
+ clearCart(Integer): void
+ modifyCart(List<ModifyCartDto>): Map<Product, Integer>
ProductController
- productService: ProductService
+ getProductDetails(Integer): ProductDto
UpdateProductHelperService
~ product: ProductDto
+ getProduct(): ProductDto
+ setProduct(ProductDto): void
VendorController
- vendorService: VendorService
+ addProduct(ProductDto): Integer
+ updateProduct(ProductDto): Integer
+ getMyProducts(Integer): List<ProductDto>
CatalogController
- vendorService: VendorService
- customerService: CustomerService
+ getProducts(): List<ProductDto>
+ getCurrentVendorProducts(Integer): List<ProductDto>
AuthService
~ imgPath: String
~ authenticatedUser: AuthenticatedUser
+ getUserType(): String
+ getImgPath(): String
NewProduct
~ productName: String
~ productInfo: String
~ stock: String
~ cost: String
+ setCost(String): void
+ getProductName(): String
+ setProductInfo(String): void
+ setProductName(String): void
+ getCost(): String
+ setStock(String): void
+ getProductInfo(): String
LoginController
- loginService: LoginService
+ signup(SignUpDto): UserInfo
LoginView
- authenticatedUser: AuthenticatedUser
~ lf: LoginForm
+ beforeEnter(BeforeEnterEvent): void
MainLayout
- appName: H1
- authenticatedUser: AuthenticatedUser
- accessChecker: AccessAnnotationChecker
- createFooter(): Footer
- getCurrentPageTitle(): String
# afterNavigation(): void
NewUser
~ mailId: String
~ shippingAddress: String
~ userType: String
~ firstName: String
~ userName: String
~ password: String
~ lastName: String
+ setUserType(String): void
+ getFirstName(): String
+ setMailId(String): void
+ getShippingAddress(): String
+ getPassword(): String
+ setShippingAddress(String): void
+ setFirstName(String): void
+ getLastName(): String
+ setLastName(String): void
+ setUserName(String): void
+ setPassword(String): void
+ getUserType(): String
+ getMailId(): String
ProductDto
- vendorId: Integer
- id: Integer
- productInfo: String
- stock: Integer
- discount: Integer
- productName: String
- cost: Double
- productImagePath: String
+ getProductInfo(): String
+ getId(): Integer
+ getProductName(): String
+ setProductImagePath(String): void
+ toString(): String
+ equals(Object): boolean
+ getVendorId(): Integer
+ getStock(): Integer
+ getCost(): Double
+ setProductInfo(String): void
+ setProductName(String): void
+ builder(): ProductDtoBuilder
+ getDiscount(): Integer
+ setStock(Integer): void
+ hashCode(): int
+ setId(Integer): void
+ getProductImagePath(): String
+ setVendorId(Integer): void
+ setDiscount(Integer): void
# canEqual(Object): boolean
+ setCost(Double): void
UserInfoDto
- mailId: String
- lastName: String
- firstName: String
- shippingAddress: byte[]
- id: Integer
+ getMailId(): String
+ getId(): Integer
+ getFirstName(): String
+ equals(Object): boolean
+ setShippingAddress(byte[]): void
+ getLastName(): String
# canEqual(Object): boolean
+ setMailId(String): void
+ builder(): UserInfoDtoBuilder
+ hashCode(): int
+ getShippingAddress(): byte[]
+ setId(Integer): void
+ setFirstName(String): void
+ toString(): String
ModifyCartDto
- userId: Integer
- productId: Integer
- quantity: Integer
+ getProductId(): Integer
+ getUserId(): Integer
+ getQuantity(): Integer
+ setProductId(Integer): void
+ toString(): String
+ setUserId(Integer): void
+ setQuantity(Integer): void
+ builder(): ModifyCartDtoBuilder
+ equals(Object): boolean
# canEqual(Object): boolean
+ hashCode(): int
Cart
- id: CartId
- quantity: Integer
+ getId(): CartId
+ setId(CartId): void
+ getQuantity(): Integer
CartId
- serialVersionUID: long
- productId: Integer
- userId: Integer
+ getProductId(): Integer
+ setUserId(Integer): void
+ hashCode(): int
+ getUserId(): Integer
+ setProductId(Integer): void
HomepageView
OrderService
- orderRepository: OrderRepository
- productRepository: ProductRepository
- orderDetailRepository: OrderDetailRepository
+ getOrders(Integer): List<Order>
- saveOrderDetail(Order, Map<Product, Integer>): void
- saveOrder(Integer, Map<Product, Integer>): Order
+ getOrder(Integer): Map<Product, Integer>
+ placeOrder(Integer, Map<Product, Integer>): void
CartService
- productRepository: ProductRepository
- cartRepository: CartRepository
+ clearCart(Integer): void
+ modifyCart(List<ModifyCartDto>): Map<Product, Integer>
ProductService
- productRepository: ProductRepository
+ getProductDetails(Integer): ProductDto
+ convertProductToProductDto(Product): ProductDto
Application
+ main(String[]): void
VendorService
- productRepository: ProductRepository
+ addProduct(ProductDto): Integer
+ updateProduct(ProductDto): Integer
+ getMyProducts(Integer): List<ProductDto>
CustomerService
- userRepository: UserRepository
- productRepository: ProductRepository
- userInfoRepository: UserInfoRepository
+ getAllProducts(): List<ProductDto>
LoginService
- userRepository: UserRepository
- userInfoRepository: UserInfoRepository
+ signup(SignUpDto): UserInfo
AuthenticatedUser
- userRepository: UserRepository
- getAuthentication(): Optional<Authentication>
+ logout(): void
+ get(): Optional<User>
UserDetailsServiceImpl
- userRepository: UserRepository
+ loadUserByUsername(String): UserDetails
UserService
- repository: UserRepository
+ update(User): User
+ count(): int
+ delete(Integer): void
+ list(Pageable): Page<User>
Order
- orderDate: Instant
- id: Integer
- userId: Integer
- totalCost: Double
+ getUserId(): Integer
+ getId(): Integer
+ getOrderDate(): Instant
+ setOrderDate(Instant): void
+ getTotalCost(): Double
+ setTotalCost(Double): void
+ setUserId(Integer): void
VendorHomepageRenderer
+ renderHomepage(AuthService, CatalogController): Component
CustomerHomepageRenderer
+ renderHomepage(AuthService, CatalogController): Component
ProductCatalogView
<<interface>>
OrderRepository
+ findAllByUserIdOrderByOrderDateDesc(Integer): List<Order>
<<interface>>
OrderDetailRepository
+ findAllById_OrderId(Integer): List<OrderDetail>
OrderDetail
- id: OrderDetailId
- quantity: Integer
+ getId(): OrderDetailId
+ setQuantity(Integer): void
+ setId(OrderDetailId): void
<<interface>>
CartRepository
+ findAllById_UserIdAndQuantityNot(Integer, Integer): List<Cart>
+ deleteAllById_UserId(Integer): void
<<interface>>
ProductRepository
+ getProductById(Integer): Product
+ getAllBy(): List<Product>
+ getProductsByVendorId(Integer): List<Product>
Product
- vendorId: Integer
- cost: Double
- productImagePath: byte[]
- productName: String
- productInfo: byte[]
- id: Integer
- discount: Integer
- stock: Integer
+ setProductImagePath(byte[]): void
+ setVendorId(Integer): void
+ getProductInfo(): byte[]
+ getStock(): Integer
+ setDiscount(Integer): void
+ setId(Integer): void
+ getVendorId(): Integer
+ getCost(): Double
+ getDiscount(): Integer
+ setProductName(String): void
+ setProductInfo(byte[]): void
+ setCost(Double): void
+ getProductImagePath(): byte[]
+ getId(): Integer
+ setStock(Integer): void
<<interface>>
UserInfoRepository
<<interface>>
UserRepository
+ findByUserName(String): User
CardLayoutHelper
+ createColumn(ProductDto, String): Component
- createCard(ProductDto, String): Component
<<interface>>
RenderBehavior
+ renderHomepage(AuthService, CatalogController): Component
OrderDetailId
- productId: Integer
- orderId: Integer
- serialVersionUID: long
+ getOrderId(): Integer
+ setOrderId(Integer): void
+ equals(Object): boolean
+ hashCode(): int
+ getProductId(): Integer
UserTypeService
~ userTypes: List<String>
+ getUserTypes(): List<String>
«create»
1 orderController
1 cartController
1 productController
1 updateProductHelperService
1 product
1 authService
1 newProduct
1 loginController
1 newUser
1 orderService
1 cartService
1 productService
1 vendorService
1 customerService
1 loginService
1 authenticatedUser
1 orderRepository
1 orderDetailRepository
1 cartRepository
1 productRepository
1 userInfoRepository
1 userRepository
1 repository
1 id

# UML Class Diagram from Project 5



## Design Patterns Used

- During Login, based on the type of user(Customer or Vendor), the Login method renders a Product Catalog page for that type of User. This uses the Strategy pattern by having multiple algorithms for login.
- The loading of the Homepage uses Composite pattern as the page loads it's child components such as list of products, the cart icon, the user icon.
- In the Product Catalog page, we use Command pattern to execute different actions for user clicks on different parts of the page such as clicking on a product, clicking on order history, clicking on cart icon, clicking on sign out, etc.
- The PlainTextPasswordEncoder Class is an implementation of Singleton Pattern.
- Click Listeners are an example of Observer pattern as the Event Handler listens for the click event to perform the defined task.

## Changes to Class Diagram from Project 5

- Since the homepage for Customer and Vendor views were similar, we used a common Controller called CatalogController to carry out the functionality required to display the homepages for both views.
- We added a CardLayoutHelper class to render the cards for the Product Catalog pages.
- We removed the Favorites feature related classes as that was not implemented.
- We added an OrderHistoryView class to display the list of orders placed by a customer.
- We modified a few of the entity and DTO Objects as required by the APIs.

## Third-Party code vs. Original code Statement

- We used the Vaadin framework for UI development.
- We used the following links for getting started with Vaadin :
  https://vaadin.com/docs/latest/tutorial
  https://start.vaadin.com/app

- We used the following links for getting started with the server-side setup:

  https://www.baeldung.com/persistence-with-spring

  https://www.baeldung.com/rest-api-spring-guide

  https://start.spring.io/
- We used Spring Security for authentication of Users.
- We used the following links as reference while implementing the UI for our pages:
  - https://www.w3schools.com/howto/howto_css_cards.asp
  - https://www.w3schools.com/howto/howto_css_column_cards.asp
  - https://designmodo.com/create-product-page/

### Statement on the OOAD process for your overall Semester Project

1. Our initial design of the system was very well-defined and robust which made it easy to implement our features once we were past the learning stage.
2. The use of Class Diagram, Database Design and UI mock-ups to define the system helped have a clear strategy for implementation.
3. Our plan to use a completely new Framework led to the miscalculation of the effort required as the framework involved a steep learning curve.