# Ecommerce

Student name: Buyanjargal Byambajav

Supervisor: Mrad Mohamed Azouz

## About the project

### Description

This project is an e-commerce platform designed as an online marketplace where users can register either as vendors or customers. It encompasses the fundamental functionalities typically found in digital commerce systems, including user registration and authentication, product listing and management by vendors, and the ability for customers to browse and purchase products. The platform aims to simulate a real-world marketplace environment, enabling seamless interaction between sellers and buyers.

### Technologies used

The project was implemented using the Java programming language, utilizing the Jmix framework(The Jmix framework is built on top of Spring and other mainstream Java technologies.) for application development. Gradle was used as the build automation tool to manage project dependencies and configurations. MySQL served as the database management system for storing and handling the application's data.

## Use-case diagram

### Description

Actors:

* Customer:
  + View, filter, sort products.
  + Add/remove products from cart.
  + Checkout cart (with payment selection).
  + View order history.
* Vendor:
  + Deploy new product (requires admin approval).
  + View, filter, sort products.
  + View received orders.
* Admin:
  + Inherits all Customer/Vendor actions.
  + Approve/reject products.
  + Add categories.

A diagram of a diagram

AI-generated content may be incorrect.

### UML class diagram

A screenshot of a computer program

AI-generated content may be incorrect.

### Class description

1.UserDetailView

* Purpose**:** Manages user detail display and editing.
* Attributes**:**
  + passwordField, confirmPasswordField: For entering and confirming passwords.
  + entityStates, usernameField, emailField, etc.: Form input fields for user attributes.
* Methods**:**
  + onBeforeSave: before saving the user entity encodes their password.
  + onValidation, onInitEntity, onReady: set properties of the text fields, set focus component, compare the passwords written.

2.MainView

* Purpose**:** The main dashboard or landing view.
* Attributes**:**
  + Multiple productDc, productDl: Data containers and loaders for bestselling products (3 products that have the most units sold).
* Methods**:**
  + onProductItemClick, onViewFirstClick, etc.: Event handlers for the view product buttons that redirect to the product detail view of the corresponding product.
  + onBeforeShow: sets the bestselling products to its instance container.

3.ProductDetailView

* Purpose**:** Handles product image uploads and details.
* Attributes**:**
  + imageField: Image of the product field.
  + notifications: UI feedback handler.
* Methods**:**
  + onFileFieldUploadSucceeded, onFileFieldUploadFailed: Handlers for upload success/failure of the product image and saves the file reference in the database.
  + onSaveAction: sets vendor field of product prior to saving the product.

4. ProductRepository

* Purpose**:** Repository interface to fetch product data.
* Methods**:**
  + findByVendor(Vendor): Fetches product list associated with a vendor.

5. DatabaseUserRepository

* Purpose**:** Custom user repository.
* Methods**:**
  + getUserClass(), initAnonymousUser(): Manage anonymous or class-specific user details.

6. EcommerceApplication

* Purpose**:** Entry point and configuration manager of the application.
* Attributes/Methods**:**
  + main(): Main method to launch the Spring Boot app.
  + Holds references to environment and data sources.

7. MyOrderListView

* Purpose**:** Displays a customer's order history.
* Methods**:**
  + onCartsDataGridItemClick(): event handler that redirects to the product detail view of the clicked product.

8. OrderItemsQuery

* Purpose**:** It has a static attribute called cartId that is used to set query parameter when displaying order items relating to a particular order.
* Attributes**:**
  + cartId: UUID identifier for the cart clicked.
* Methods**:**
  + setCartId, getCartId: Accessor methods for cart filtering.

9. ProfileView

* Purpose**:** Displays user profile details.
* Attributes**:**
  + userDc, profileDl: Container of data collection and data loader for user data.
* Methods**:**
  + onBeforeShow: sets the currect user’s data to the containers..

10. VendorOrderItemListView

* **Purpose:** Displays list of orders of the products that a currently logged in vendor..

11. UserListView

* **Purpose:** Admin view for listing all registered users.

12. CategoryDetailView

* **Purpose:** View for managing an entity of category.

13. CategoryListView

* **Purpose:** Lists all categories (for admin).

14. RequestedProductListView

* Purpose**:** Displays list of products submitted by vendors pending approval.
* Methods**:**
  + onProductsDataGridItemClick(): Triggers when a product is clicked, navigates to product detail view corresponds to the clicked product.

15. AllProductListView

* Purpose**:** Displays all available products on the platform.
* Methods**:**
  + onProductsDataGridItemClick():Triggered when a product is clicked, navigates to product detail view corresponds to the clicked product.

16. RequestedProductDetailView

* Purpose**:** Admin interface to accept or reject product submissions.
* Methods**:**
  + onRejectButtonClick, onAcceptButtonClick: event handles for the buttons approve/reject, sets acceptance attribute of the product to the chosen enum.

17. LoginView

* Purpose**:** User login interface.
* Attributes**:**
  + defaultUsername, defaultPassword, loginField: Fields for authentication.
* Methods**:**
  + onLogin(), onInit(): Handle login process.
  + onRegisterButtonClick: navigates to user registration view.
  + initDefaultCredentials: sets default username and password.

18. RegistrationService

* Purpose**:** Handles user registration.
* Methods**:**
  + registerCustomer, registerVendor: registers a user and assigns corresponding access role.

19. CartService

* Purpose**:** Manages cart operations.
* Methods**:**
  + processCheckout(): Finalize the checkout, changes necessary attributes of the cart that is being checked out, also sets necessary attributes of cart items in it, depending on the payment type the method sets user balance as well.
  + getOrCreateCart(): checks if a user has ‘not checked out’ cart (cart), if so returns the card, if not creates a new cart.
  + addToCart(): add a particular number of products to a cart.
  + calculateTotal(): calculates total amount of items in a cart.

20. MyCartItemListView

* Purpose**:** Displays current user's cart items.
* Attributes**:**
  + productCartItemsDc: Data container for cart items.
* Methods**:**
  + onCheckOutButtonClick(): event handler for checking out, shows a dialog for choosing payment type.
  + onRemoveButtonClick(): event handler for removing a product from a cart.

21. ProductShowView

* Purpose**:** Displays individual product details for customers.
* Attributes**:**
  + productDc: Product data container.
* Methods**:**
  + onAddToCartButtonClick(): Adds item to the cart, it shows an input dialog for user to indicate how many units of the product they want to add to their cart, shows error message if requested amount exceeds stock .

22. OrderItemListView

* Purpose**:** Displays order items.
* Attributes**:**
  + productCartItemsDc: Cart-related product items.
* Methods:
  + onInit: sets data containers of the view to the order items that correcponds to a particular order(cart that is checked out).

23. UserRegisterView

* Purpose**:** Interface for registering new users.
* Attributes**:**
  + firstNameField, lastNameField, emailField, usernameField, passwordField, vendorCheckBox: Fields for user input.
* Methods**:**
  + onRegisterButtonClick, navigateToLoginView: register a new user, and sets iits attributes with the user input, and navigates to the log in view.

### Database description

1. User

Represents any type of user in the system: customer, vendor, or admin.

* Fields**:**
  + id (UUID): Primary key.
  + username (String): User's login name.
  + password (String): Encrypted user password.
  + email (String): User email.
  + firstName, lastName (String): name of the user.
  + balance (Double): User's account balance that they need to pay.
  + active (Boolean): Indicates if the user is active.
  + authorities (Collection<GrantedAuthority>): Roles/permissions assigned to the user.
* Relationships**:**
  + One-to-many with Product (as vendor).
  + One-to-many with Cart (as customer).

1. Product

Represents a product listed for sale on the platform.

* Fields**:**
  + id (UUID): Primary key.
  + name (String): Product name.
  + image (FileRef): Reference to product image.
  + price (Double): Unit price.
  + stock (BigInteger): Quantity available.
  + unitsSold (Integer): Units already sold.
  + isAccepted (String): Approval status (e.g., waiting, accepted, rejected).
* Relationships**:**
  + Many-to-one with User (vendor).
  + Many-to-one with Category.
  + One-to-many with ProductCartItem.

1. Category

Defines product categories.

* Fields**:**
  + id (UUID): Primary key.
  + name (String): Category name.
* Relationships**:**
  + One-to-many with Product.

1. Cart

Represents a customer’s cart, which may be open(not check out) or checked out.

* Fields**:**
  + id (UUID): Primary key.
  + total (Double): Total amount for all items.
  + status (Status): Status of the cart (e.g., notCheckedOut, CheckedOut).
  + checkedOutDate (LocalDate): When the cart was finalized(checked out).
  + customerName (String): name of the customer whom the cart belong to.
  + paymentType (String): Payment method selected.
* Relationships**:**
  + Many-to-one with User (customer).
  + Many-to-one with Status.
  + One-to-many with ProductCartItem.

1. ProductCartItem

Represents a product entry in a cart.

* Fields**:**
  + id (UUID): Primary key.
  + amount (Double): Total cost of the item in the cart.
  + unit (Integer): Quantity of the product.
  + paymentType (String): Chosen payment method.
  + date (LocalDate): Date added to cart.
* Relationships**:**
  + Many-to-one with Product.
  + Many-to-one with Cart.

1. Status

Used to define different states for a cart.

* Fields**:**
  + id (UUID): Primary key.
  + name (String): Status name.
* Relationships**:**
  + One-to-many with Cart.

### Entity diagram

A screenshot of a computer

AI-generated content may be incorrect.