CENG201 PROJECT

Generated by Doxygen 1.12.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Ui Namespace Reference

Contains the UI class for the login window.

5.1.1 Detailed Description

Contains the UI class for the login window.

Chapter 6

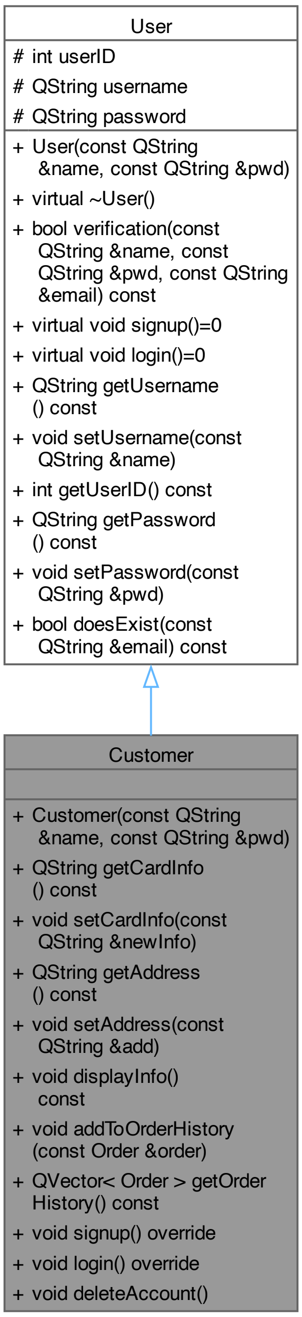
Class Documentation

6.1 Customer Class Reference

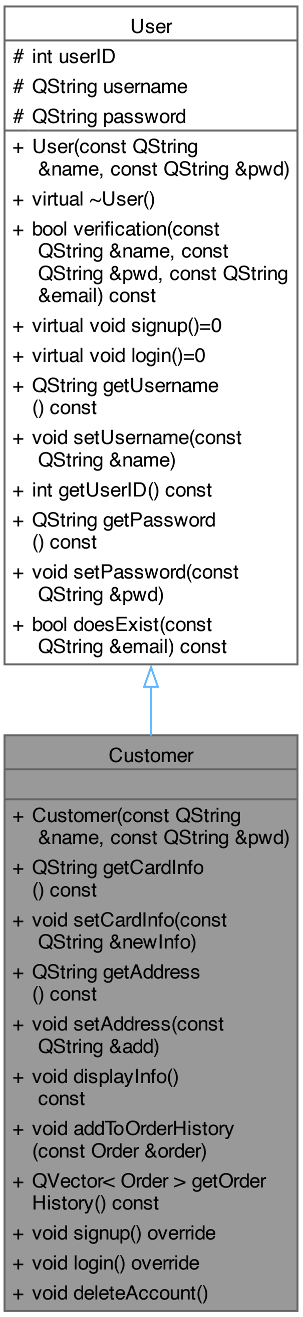
Represents a customer inheriting from the User class.

```
#include <Customer.h>
```

Inheritance diagram for Customer:


User
# int userID
# QString username
# QString password
+ User(const QString &name, const QString &pwd)
+ virtual ~User()
+ bool verification(const QString &name, const QString &pwd, const QString &email) const
+ virtual void signup()=0
+ virtual void login()=0
+ QString getUsername () const
+ void setUsername(const QString &name)
+ int getUserID() const
+ QString getPassword () const
+ void setPassword(const QString &pwd)
+ bool doesExist(const QString &email) const
Customer
+ Customer(const QString &name, const QString &pwd)
+ QString getCardInfo () const
+ void setCardInfo(const QString &newInfo)
+ QString getAddress () const
+ void setAddress(const QString &add)
+ void displayInfo() const
+ void addToOrderHistory (const Order &order)
+ QVector< Order > getOrder History() const
+ void signup() override
+ void login() override
+ void deleteAccount()

Collaboration diagram for Customer:


User
# int userID
# QString username
# QString password
+ User(const QString &name, const QString &pwd)
+ virtual ~User()
+ bool verification(const QString &name, const QString &pwd, const QString &email) const
+ virtual void signup()=0
+ virtual void login()=0
+ QString getUsername () const
+ void setUsername(const QString &name)
+ int getUserID() const
+ QString getPassword () const
+ void setPassword(const QString &pwd)
+ bool doesExist(const QString &email) const
Customer
+ Customer(const QString &name, const QString &pwd)
+ QString getCardInfo () const
+ void setCardInfo(const QString &newInfo)
+ QString getAddress () const
+ void setAddress(const QString &add)
+ void displayInfo() const
+ void addToOrderHistory (const Order &order)
+ QVector< Order > getOrder History() const
+ void signup() override
+ void login() override
+ void deleteAccount()

Public Member Functions

- Customer (const QString &name, const QString &pwd)

  Constructor for the Customer class.
- QString getCardInfo () const

  Gets the customer's card information.
- void setCardInfo (const QString &newInfo)

Sets new card information for the customer.
- QString getAddress () const

  Gets the customer's address.
- void setAddress (const QString &add)

  Sets the customer's address.
- void displayInfo () const

  Displays customer information.
- void addToOrderHistory (const Order &order)

  Adds a completed order to the customer's order history.
- QVector< Order > getOrderHistory () const

  Gets the customer's order history.
- void signup () override

  Overridden signup function from the User class.
- void login () override

  Overridden login function from the User class.
- void deleteAccount ()

  Deletes the customer's account.

Public Member Functions inherited from User

- User (const QString &name, const QString &pwd)

  Constructs a User object.
- virtual ∼User ()

  Virtual destructor for the User class.
- bool verification (const QString &name, const QString &pwd, const QString &email) const

  Verifies the user's credentials.
- QString getUsername () const

  Retrieves the username of the user.
- void setUsername (const QString &name)

  Sets a new username for the user.
- int getUserID () const

  Retrieves the user ID.
- QString getPassword () const

  Retrieves the user's password.
- void setPassword (const QString &pwd)

  Sets a new password for the user.
- bool doesExist (const QString &email) const

  Checks if the given email exists in the user data file.

Additional Inherited Members

Protected Attributes inherited from User

- int **userID**

  Unique identifier for the user.
- QString **username**

  The username of the user.
- QString **password**

  The password of the user.

6.1.1 Detailed Description

Represents a customer inheriting from the User class.

Derived class representing a customer in the system.

The Customer class provides additional functionality for handling customer-specific attributes like card information and address. It also supports signup and login functionalities.

The Customer class extends the User class by adding additional functionality specific to customers, such as storing card information, delivery address, and order history. It also provides methods for signing up, logging in, and managing customer-specific data.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Customer()

```
Customer::Customer (
             const QString & name,
             const QString & pwd)
```

Constructor for the Customer class.

Constructs a new Customer object with the given name and password.

Initializes a new Customer object with the given username and password.

Parameters

name	Username for the customer.
pwd	Password for the customer.
name	The username of the customer.
pwd	The password of the customer.

6.1.3 Member Function Documentation

6.1.3.1 addToOrderHistory()

```
void Customer::addToOrderHistory (
             const Order & order)
```

Adds a completed order to the customer's order history.

Parameters

order	The completed order.

6.1.3.2 deleteAccount()

```
void Customer::deleteAccount ()
```

Deletes the customer's account.

Removes the customer's information from the system.

6.1.3.3 displayInfo()

```
void Customer::displayInfo () const
```

Displays customer information.

Displays the customer's information including username and address.

Includes the customer's username, address, and the size of their order history.

6.1.3.4 getAddress()

```
QString Customer::getAddress () const
```

Gets the customer's address.

Retrieves the customer's address.

Returns

The address as a QString.

QString The address.

6.1.3.5 getCardInfo()

```
QString Customer::getCardInfo () const
```

Gets the customer's card information.

Retrieves the customer's card information.

Returns

The card information as a QString.

QString The card information.

6.1.3.6 getOrderHistory()

```
QVector< Order > Customer::getOrderHistory () const
```

Gets the customer's order history.

Returns

A QVector of Order objects representing the customer's past orders.

6.1.3.7 login()

```
void Customer::login () [override], [virtual]
```

Overridden login function from the User class.

Logs in an existing customer by verifying their credentials in a file.

Verifies the customer's credentials and logs them into the system.

This method checks if the provided username and password match an entry in the file.

Implements User.

6.1.3.8 setAddress()

```
void Customer::setAddress (
             const QString & add)
```

Sets the customer's address.

Updates the customer's address.

Parameters

add	The new address to set.
add	The new address to be set.

6.1.3.9 setCardInfo()

```
void Customer::setCardInfo (
             const QString & newInfo)
```

Sets new card information for the customer.

Updates the customer's card information.

Parameters

newInfo	The new card information to set.
newInfo	The new card information to be set.

6.1.3.10 signup()

```
void Customer::signup ()  [override], [virtual]
```

Overridden signup function from the User class.

Registers a new customer by appending their details to a file.

Handles the process of registering a new customer in the system.

This method checks if the username and password already exist in the file. If not, it appends the new user details to the file.

Implements User.

The documentation for this class was generated from the following files:

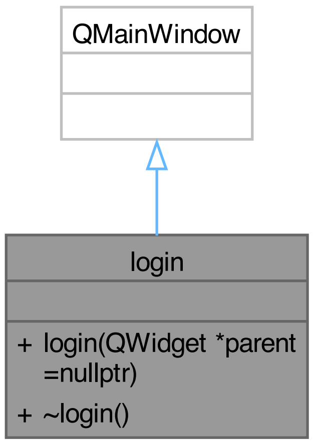
- Customer.h
- Customer.cpp

6.2 login Class Reference

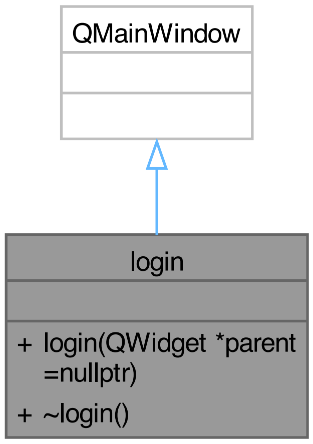
Handles user login and signup functionalities in the application.

```
#include <login.h>
```

Inheritance diagram for login:



Collaboration diagram for login:



Public Member Functions

- login (QWidget ∗parent=nullptr)

  Constructor for the login class.
- ∼login ()

  Destructor for the login class.

6.2.1 Detailed Description

Handles user login and signup functionalities in the application.

Handles the user interface and logic for login and signup.

The login class provides the user interface and backend logic for handling user authentication and account creation. It interacts with the Customer class to validate user credentials and store new user data.

The `login` class provides functionalities for user authentication and account creation. It includes slots for handling login and signup actions and integrates with the `Customer` class for backend user management.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 login()

```
login::login (
            QWidget ∗ parent = nullptr)
```

Constructor for the login class.

Initializes the login window and its UI components.

Parameters

parent	The parent widget, default is nullptr.

Initializes the UI components for the login window.

Parameters

parent	Parent widget for the login window.

6.2.2.2 ∼login()

```
login::∼login ()
```

Destructor for the login class.

Cleans up dynamically allocated UI resources.

Cleans up the UI resources.

The documentation for this class was generated from the following files:

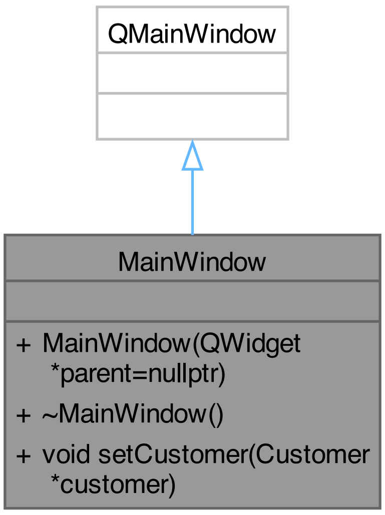
- login.h
- login.cpp

6.3 MainWindow Class Reference

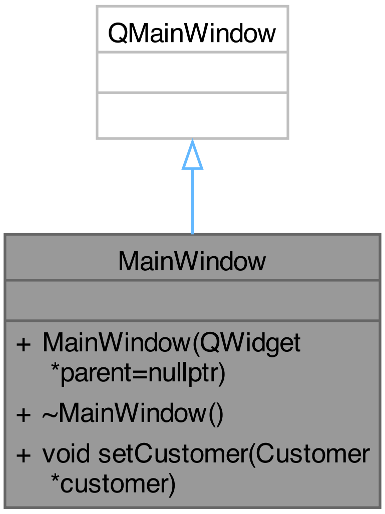
Represents the main application window for browsing and purchasing products.

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:


QMainWindow
MainWindow
+ MainWindow(QWidget *parent=nullptr)
+ ~MainWindow()
+ void setCustomer(Customer *customer)

Collaboration diagram for MainWindow:


QMainWindow
MainWindow
+ MainWindow(QWidget *parent=nullptr)
+ ~MainWindow()
+ void setCustomer(Customer *customer)

Public Member Functions

- MainWindow (QWidget ∗parent=nullptr)

  Constructs a new MainWindow object.
- ∼MainWindow ()

  Destroys the MainWindow object.
- void setCustomer (Customer ∗customer)

  Sets the customer for the main window.

6.3.1 Detailed Description

Represents the main application window for browsing and purchasing products.

Represents the main application window for the e-commerce application.

The `MainWindow` class handles the display of available products, cart management, discount application, and order confirmation. It interacts with the `Customer` and `Order` classes to facilitate e-commerce functionalities.

The `MainWindow` class provides functionalities for browsing products, managing a shopping cart, applying discounts, and confirming orders. It interacts with the `Customer`, `Order`, and `Product` classes to manage the e-commerce workflow.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 MainWindow()

```
MainWindow::MainWindow (
            QWidget * parent = nullptr)  [explicit]
```

Constructs a new MainWindow object.

Constructor for the MainWindow class.

Initializes the UI components and sets up product and cart management.

Parameters

parent	The parent widget, default is nullptr.

Initializes the UI components, populates the product list, and connects signals to slots.

Parameters

parent	The parent widget, default is nullptr.

6.3.2.2 ∼MainWindow()

```
MainWindow::∼MainWindow ()
```

Destroys the MainWindow object.

Destructor for the MainWindow class.

Cleans up dynamically allocated UI resources.

6.3.3 Member Function Documentation

6.3.3.1 setCustomer()

```
void MainWindow::setCustomer (
            Customer * customer)
```

Sets the customer for the main window.

Parameters

customer	Pointer to the Customer object.

The documentation for this class was generated from the following files:

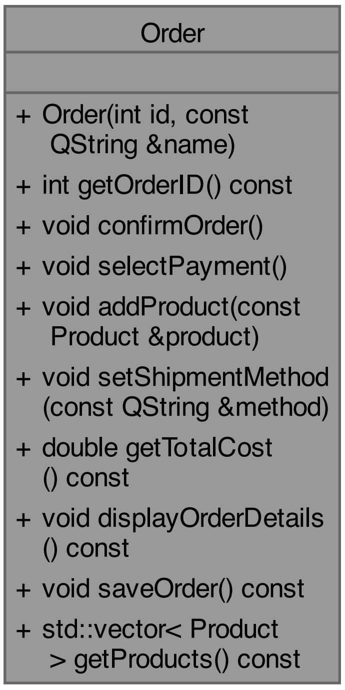
- mainwindow.h
- mainwindow.cpp

6.4 Order Class Reference

Represents a customer's order, containing products and shipment details.

```
#include <order.h>
```

Collaboration diagram for Order:


Order
+ Order(int id, const QString &name)
+ int getOrderID() const
+ void confirmOrder()
+ void selectPayment()
+ void addProduct(const Product &product)
+ void setShipmentMethod (const QString &method)
+ double getTotalCost () const
+ void displayOrderDetails () const
+ void saveOrder() const
+ std::vector< Product > getProducts() const

Public Member Functions

- Order (int id, const QString &name)

  Constructor for the Order class.
- int getOrderID () const

  Retrieves the unique ID of the order.
- void confirmOrder ()

  Confirms the order by finalizing its details.
- void selectPayment ()

  Allows the customer to select a payment method for the order.
- void addProduct (const Product &product)

  Adds a product to the order.
- void setShipmentMethod (const QString &method)

  Sets the shipment method for the order.
- double getTotalCost () const

  Calculates the total cost of the order.

- void displayOrderDetails () const

  Displays all the details of the order.
- void saveOrder () const

  Saves the order details to a file or database.
- std::vector< Product > getProducts () const

  Retrieves the list of products in the order.

6.4.1 Detailed Description

Represents a customer's order, containing products and shipment details.

Represents an order placed by a customer, containing products and shipment details.

The `Order` class manages the products in an order, calculates the total cost, and handles shipment and payment-related functionalities.

The `Order` class manages the list of products, shipment details, and the total cost for an order. It also allows saving order details and selecting a payment method.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Order()

```
Order::Order (
             int id,
             const QString & name)
```

Constructor for the Order class.

Constructs an Order object.

Initializes the order with a unique ID and customer name.

Parameters

id	Unique order ID.
name	Name of the customer placing the order.
id	The unique identifier for the order.
name	The name of the customer associated with the order.

6.4.3 Member Function Documentation

6.4.3.1 addProduct()

```
void Order::addProduct (
             const Product & product)
```

Adds a product to the order.

Parameters

product	The `Product` object to add to the order.
product	The `Product` object to be added to the order.

6.4.3.2 confirmOrder()

```
void Order::confirmOrder ()
```

Confirms the order by finalizing its details.

Confirms the order.

This method should be called when the order is ready to be processed.

This method is called when the customer finalizes the order.

6.4.3.3 displayOrderDetails()

```
void Order::displayOrderDetails () const
```

Displays all the details of the order.

Displays the details of the order.

Prints information such as the order ID, customer name, list of products, shipping method, and total cost.

Prints the order ID, customer name, list of products, shipment method, and total cost.

6.4.3.4 getOrderID()

```
int Order::getOrderID () const
```

Retrieves the unique ID of the order.

Retrieves the unique identifier of the order.

Returns

The order ID as an integer.

6.4.3.5 getProducts()

```
std::vector< Product > Order::getProducts () const
```

Retrieves the list of products in the order.

Returns

A vector of `Product` objects included in the order.

6.4.3.6 getTotalCost()

```
double Order::getTotalCost () const
```

Calculates the total cost of the order.

The total cost includes the sum of product prices and the shipping cost.

Returns

The total cost of the order as a double.

The total cost includes the sum of product prices and the shipment cost.

Returns

The total cost of the order as a double.

6.4.3.7 saveOrder()

```
void Order::saveOrder () const
```

Saves the order details to a file or database.

This method is a placeholder for order persistence logic.

6.4.3.8 selectPayment()

```
void Order::selectPayment ()
```

Allows the customer to select a payment method for the order.

Selects the payment method for the order.

This method acts as a placeholder for payment-related logic.

This method is a placeholder for payment selection logic.

6.4.3.9 setShipmentMethod()

```
void Order::setShipmentMethod (
             const QString & method)
```

Sets the shipment method for the order.

Parameters

method	The shipping method (e.g., "Standard Shipping").
method	The shipment method (e.g., "Standard Shipping").

The documentation for this class was generated from the following files:

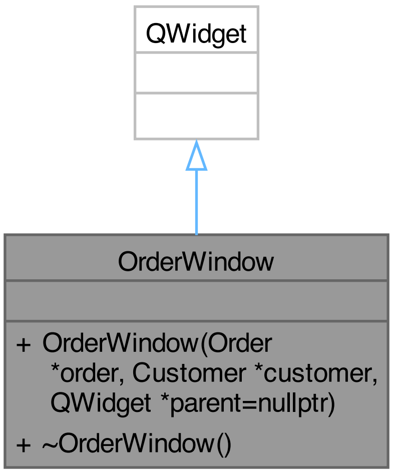
- order.h
- order.cpp

6.5 OrderWindow Class Reference

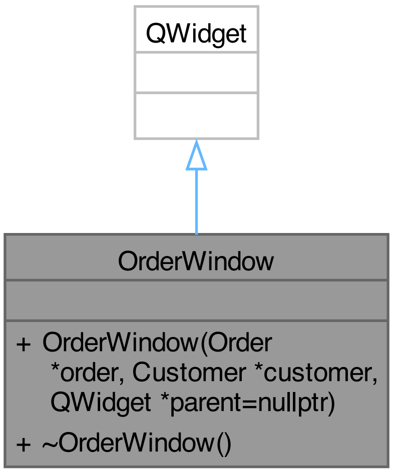
Represents the window for reviewing and confirming an order.

```
#include <orderwindow.h>
```

Inheritance diagram for OrderWindow:


QWidget
OrderWindow
+ OrderWindow(Order *order, Customer *customer, QWidget *parent=nullptr)
+ ~OrderWindow()

Collaboration diagram for OrderWindow:


QWidget
OrderWindow
+ OrderWindow(Order *order, Customer *customer, QWidget *parent=nullptr)
+ ~OrderWindow()

Public Member Functions

- OrderWindow (Order ∗order, Customer ∗customer, QWidget ∗parent=nullptr)

  Constructs an OrderWindow object.
- ∼OrderWindow ()

  Destroys the OrderWindow object.

6.5.1 Detailed Description

Represents the window for reviewing and confirming an order.

Provides a user interface for reviewing and confirming orders.

The `OrderWindow` class allows customers to review the details of their order, including products, total cost, and shipment method. It also provides options for payment and order confirmation.

The `OrderWindow` class allows customers to view the details of their order, including products, total cost, and shipment options. It also provides options for confirming the order and selecting a payment method.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 OrderWindow()

```
OrderWindow::OrderWindow (
            Order ∗ order,
            Customer ∗ customer,
            QWidget ∗ parent = nullptr)   [explicit]
```

Constructs an OrderWindow object.

Constructs the OrderWindow.

Initializes the UI components and displays the order details.

Parameters

order	Pointer to the `Order` object containing the order's details.
customer	Pointer to the `Customer` object associated with the order.
parent	The parent widget, default is nullptr.

Initializes the UI components and displays the order details.

Parameters

order	Pointer to the `Order` object containing order details.
customer	Pointer to the `Customer` object associated with the order.
parent	The parent widget, default is nullptr.

6.5.2.2 ∼OrderWindow()

```
OrderWindow::∼OrderWindow ()
```

Destroys the OrderWindow object.

Destructor for the OrderWindow class.

Cleans up dynamically allocated UI resources.

The documentation for this class was generated from the following files:

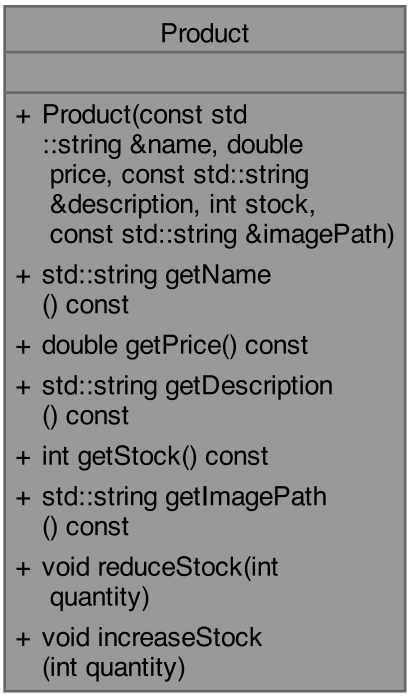
- orderwindow.h
- orderwindow.cpp

6.6 Product Class Reference

Represents a product in the system with attributes such as name, price, description, stock, and image path.

```
#include <Product.h>
```

Collaboration diagram for Product:


Product
+ Product(const std::string &name, double price, const std::string &description, int stock, const std::string &imagePath)
+ std::string getName() const
+ double getPrice() const
+ std::string getDescription() const
+ int getStock() const
+ std::string getImagePath() const
+ void reduceStock(int quantity)
+ void increaseStock(int quantity)

Public Member Functions

- Product (const std::string &name, double price, const std::string &description, int stock, const std::string &imagePath)

  Constructs a Product object.
- std::string getName () const

  Retrieves the name of the product.
- double getPrice () const

  Retrieves the price of the product.
- std::string getDescription () const

  Retrieves the description of the product.
- int getStock () const

  Retrieves the current stock quantity of the product.
- std::string getImagePath () const

  Retrieves the file path to the product's image.
- void reduceStock (int quantity)

  Reduces the stock of the product by a specified quantity.
- void increaseStock (int quantity)

  Increases the stock of the product by a specified quantity.

6.6.1 Detailed Description

Represents a product in the system with attributes such as name, price, description, stock, and image path.

The `Product` class provides methods to access product details, adjust stock, and retrieve image paths.

The `Product` class provides methods for accessing product details, managing stock levels, and retrieving the associated image path.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Product()

```
Product::Product (
            const std::string & name,
            double price,
            const std::string & description,
            int stock,
            const std::string & imagePath)
```

Constructs a Product object.

Initializes the product with the specified name, price, description, stock quantity, and image path.

Parameters

name	The name of the product.
price	The price of the product.
description	A brief description of the product.
stock	The initial stock quantity of the product.
imagePath	The file path to the product's image.

6.6.3 Member Function Documentation

6.6.3.1 getDescription()

```
std::string Product::getDescription () const
```

Retrieves the description of the product.

Returns

The description of the product as a std::string.

6.6.3.2 getImagePath()

```
std::string Product::getImagePath () const
```

Retrieves the file path to the product's image.

Returns

The image path as a std::string.

6.6.3.3 getName()

```
std::string Product::getName () const
```

Retrieves the name of the product.

Returns

The name of the product as a std::string.

6.6.3.4 getPrice()

```
double Product::getPrice () const
```

Retrieves the price of the product.

Returns

The price of the product as a double.

6.6.3.5 getStock()

```
int Product::getStock () const
```

Retrieves the current stock quantity of the product.

Returns

The stock quantity as an integer.

6.6.3.6 increaseStock()

```
void Product::increaseStock (
             int quantity)
```

Increases the stock of the product by a specified quantity.

Parameters

quantity	The quantity to add to the stock.

6.6.3.7 reduceStock()

```
void Product::reduceStock (
            int quantity)
```

Reduces the stock of the product by a specified quantity.

Ensures that the stock is not reduced below zero.

Parameters

quantity	The quantity to reduce from the stock.

The documentation for this class was generated from the following files:

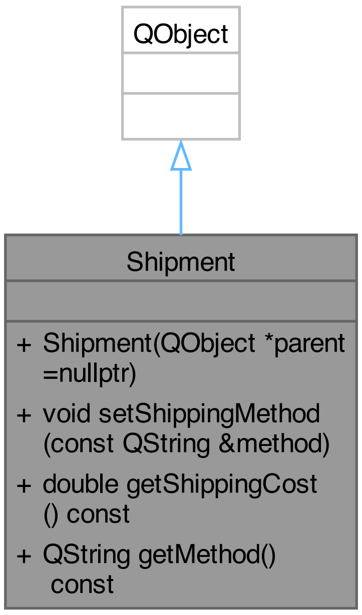
- Product.h
- Product.cpp

6.7 Shipment Class Reference

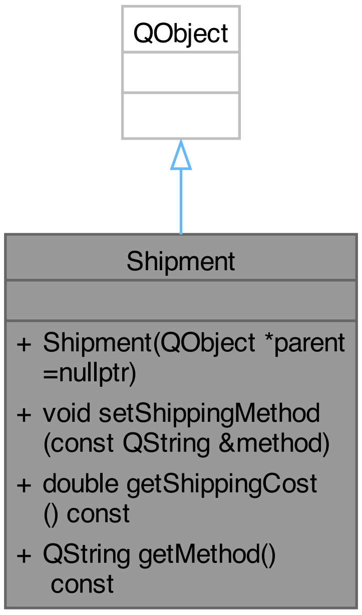
Manages shipment details, including the method and associated cost.

`#include <Shipment.h>`

Inheritance diagram for Shipment:


QObject
Shipment
+ Shipment(QObject *parent =nullptr)
+ void setShippingMethod (const QString &method)
+ double getShippingCost () const
+ QString getMethod() const

Collaboration diagram for Shipment:


QObject
Shipment
+ Shipment(QObject *parent =nullptr)
+ void setShippingMethod (const QString &method)
+ double getShippingCost () const
+ QString getMethod() const

Public Member Functions

- Shipment (QObject ∗parent=nullptr)

  Constructs a Shipment object.
- void setShippingMethod (const QString &method)

  Sets the shipping method and calculates the associated cost.
- double getShippingCost () const

  Retrieves the current shipping cost.
- QString getMethod () const

  Retrieves the current shipping method.

6.7.1 Detailed Description

Manages shipment details, including the method and associated cost.

Manages shipment details in the e-commerce system.

The `Shipment` class provides functionalities to set the shipping method, calculate the associated shipping cost, and retrieve the current shipping details.

The `Shipment` class allows setting a shipping method, calculating the shipping cost, and retrieving the current shipping details.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Shipment()

```
Shipment::Shipment (
             QObject * parent = nullptr)  [explicit]
```

Constructs a Shipment object.

Initializes the shipment with default values: "Standard" shipping method and its associated cost.

Parameters

parent	Pointer to the parent QObject, default is nullptr.

Initializes the shipment with default values: "Standard" shipping method and a cost of $5.0.

Parameters

parent	Pointer to the parent QObject, default is nullptr.

6.7.3 Member Function Documentation

6.7.3.1 getMethod()

```
QString Shipment::getMethod () const
```

Retrieves the current shipping method.

Returns

The shipping method as a QString.

6.7.3.2 getShippingCost()

```
double Shipment::getShippingCost () const
```

Retrieves the current shipping cost.

Retrieves the shipping cost for the current shipping method.

Returns

The shipping cost as a double.

6.7.3.3 setShippingMethod()

```
void Shipment::setShippingMethod (
             const QString & method)
```

Sets the shipping method and calculates the associated cost.

Sets the shipping method for the shipment.

Updates the shipping cost based on the provided shipping method. Supported methods include:

- "Standard": $5.0
- "Express": $10.0
- "Overnight": $20.0

If an invalid method is provided, the default method ("Standard") is applied.

Parameters

method	The selected shipping method as a QString.

Updates the shipping cost based on the selected shipping method. If an invalid method is provided, the method defaults to "Standard" with a cost of $5.0.

Parameters

method	The shipping method, e.g., "Standard", "Express", or "Overnight".

The documentation for this class was generated from the following files:

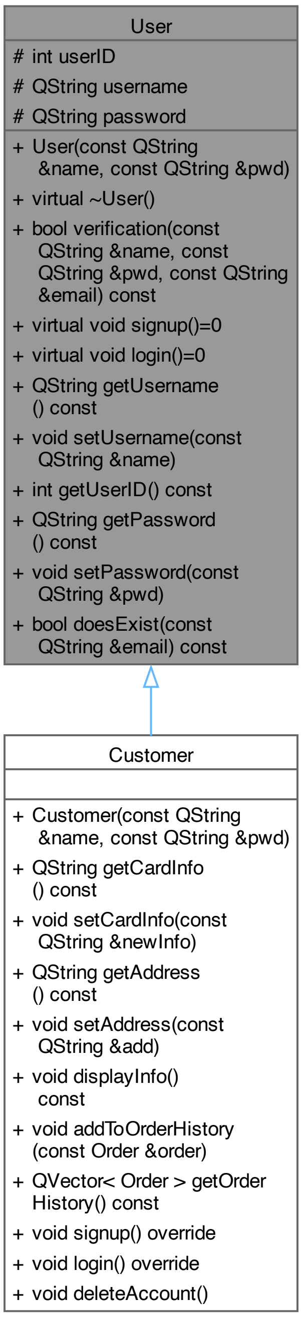
- Shipment.h
- Shipment.cpp

6.8 User Class Reference

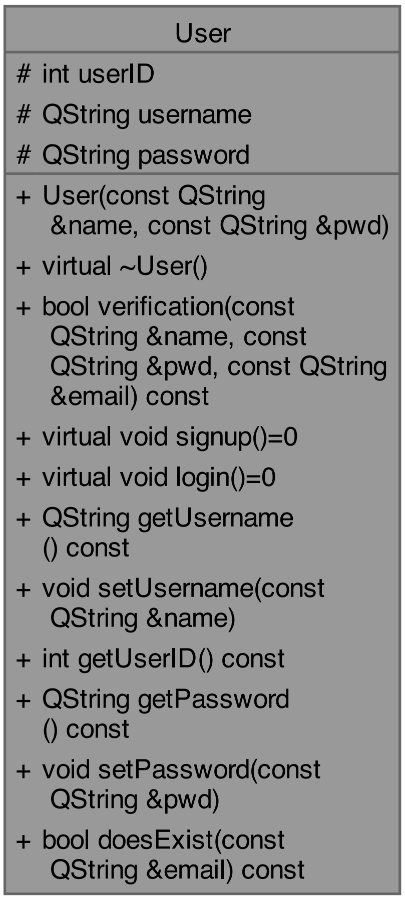
Represents a user in the system with attributes such as username, password, and user ID.

```
#include <User.h>
```

Inheritance diagram for User:


User
# int userID
# QString username
# QString password
+ User(const QString &name, const QString &pwd)
+ virtual ~User()
+ bool verification(const QString &name, const QString &pwd, const QString &email) const
+ virtual void signup()=0
+ virtual void login()=0
+ QString getUsername () const
+ void setUsername(const QString &name)
+ int getUserID() const
+ QString getPassword () const
+ void setPassword(const QString &pwd)
+ bool doesExist(const QString &email) const
Customer
+ Customer(const QString &name, const QString &pwd)
+ QString getCardInfo () const
+ void setCardInfo(const QString &newInfo)
+ QString getAddress () const
+ void setAddress(const QString &add)
+ void displayInfo() const
+ void addToOrderHistory (const Order &order)
+ QVector< Order > getOrder History() const
+ void signup() override
+ void login() override
+ void deleteAccount()

Collaboration diagram for User:


User
# int userID
# QString username
# QString password
+ User(const QString &name, const QString &pwd)
+ virtual ~User()
+ bool verification(const QString &name, const QString &pwd, const QString &email) const
+ virtual void signup()=0
+ virtual void login()=0
+ QString getUsername () const
+ void setUsername(const QString &name)
+ int getUserID() const
+ QString getPassword () const
+ void setPassword(const QString &pwd)
+ bool doesExist(const QString &email) const

Public Member Functions

- User (const QString &name, const QString &pwd)

  Constructs a User object.
- virtual ∼User ()

  Virtual destructor for the User class.
- bool verification (const QString &name, const QString &pwd, const QString &email) const

  Verifies the user's credentials.
- virtual void signup ()=0

  Virtual method for signing up.
- virtual void login ()=0

  Virtual method for logging in.
- QString getUsername () const

  Retrieves the username of the user.
- void setUsername (const QString &name)

  Sets a new username for the user.

- int getUserID () const

  Retrieves the user ID.
- QString getPassword () const

  Retrieves the user's password.
- void setPassword (const QString &pwd)

  Sets a new password for the user.
- bool doesExist (const QString &email) const

  Checks if the given email exists in the user data file.

Protected Attributes

- int **userID**

  Unique identifier for the user.
- QString **username**

  The username of the user.
- QString **password**

  The password of the user.

6.8.1 Detailed Description

Represents a user in the system with attributes such as username, password, and user ID.

Base class representing a user in the system.

The `User` class provides methods for managing user authentication, verification, and file-based existence checks.

The `User` class serves as a base for all user types, providing core attributes such as username, password, and user ID. It also defines abstract methods for signing up and logging in, which must be implemented by derived classes.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 User()

```
User::User (
             const QString & name,
             const QString & pwd)
```

Constructs a User object.

Constructs a User object with a specified username and password.

Initializes the user with a username and password. The user ID is automatically generated.

Parameters

name	Username of the user.
pwd	User's password.

Automatically generates a unique user ID.

Parameters

name	The username for the user.
pwd	The password for the user.

6.8.2.2 ∼User()

```
User::∼User ()  [virtual]
```

Virtual destructor for the User class.

Destroys the User object.

6.8.3 Member Function Documentation

6.8.3.1 doesExist()

```
bool User::doesExist (
             const QString & email) const
```

Checks if the given email exists in the user data file.

Checks if the user exists in the user data file based on the email.

Searches the user data file to verify if the specified email is already associated with an account.

Parameters

email	The email address to check.

Returns

True if the email exists; otherwise, false.

Parameters

email	The email to check for existence.

Returns

True if the email exists; otherwise, false.

6.8.3.2 getPassword()

```
QString User::getPassword () const
```

Retrieves the user's password.

Retrieves the password of the user.

Returns

The password as a QString.

6.8.3.3 getUserID()

```
int User::getUserID () const
```

Retrieves the user ID.

Returns

The user ID as an integer.

6.8.3.4 getUsername()

```
QString User::getUsername () const
```

Retrieves the username of the user.

Returns

The username as a QString.

6.8.3.5 login()

```
virtual void User::login ()  [pure virtual]
```

Virtual method for logging in.

This method must be implemented by derived classes to handle the login process.

Implemented in Customer.

6.8.3.6 setPassword()

```
void User::setPassword (
             const QString & pwd)
```

Sets a new password for the user.

Parameters

pwd	The new password to set.
pwd	The new password.

6.8.3.7 setUsername()

```
void User::setUsername (
             const QString & name)
```

Sets a new username for the user.

Parameters

name	The new username to set.
name	The new username.

6.8.3.8 signup()

```
virtual void User::signup ()  [pure virtual]
```

Virtual method for signing up.

This method must be implemented by derived classes to handle the signup process.

Implemented in Customer.

6.8.3.9 verification()

```
bool User::verification (
             const QString & name,
             const QString & pwd,
             const QString & email) const
```

Verifies the user's credentials.

Verifies if the username, password, and email combination exists in the user data file.

Checks the username, password, and email against the records in the user data file.

Parameters

name	Username to verify.
pwd	Password to verify.
email	Email to verify.

Returns

True if the credentials match; otherwise, false.

Parameters

name	The username to verify.
pwd	The password to verify.
email	The email to verify.

Returns

True if the combination exists; otherwise, false.

The documentation for this class was generated from the following files:

- User.h
- User.cpp

Chapter 7

File Documentation

7.1 Customer.h

```
#ifndef CUSTOMER_H
#define CUSTOMER_H

#include "User.h"
#include <QVector>
#include <QString>
#include "Order.h"

class Customer : public User {
private:
    QString cardInfo;
    QString address;
    QVector<Order> orderHistory;

public:
    Customer(const QString& name, const QString& pwd);

    QString getCardInfo() const;

    void setCardInfo(const QString& newInfo);

    QString getAddress() const;

    void setAddress(const QString& add);

    void displayInfo() const;

    void addToOrderHistory(const Order& order);

    QVector<Order> getOrderHistory() const;

    void signup() override;

    void login() override;

    void deleteAccount();
};

#endif // CUSTOMER_H
```

7.2 login.h

```
#ifndef LOGIN_H
#define LOGIN_H

#include <QMainWindow>
#include <QLineEdit>
#include "Customer.h"

QT_BEGIN_NAMESPACE
namespace Ui {
    class login;
}
QT_END_NAMESPACE
```

```

class login : public QMainWindow
{
    Q_OBJECT

public:
    login(QWidget *parent = nullptr);

    ~login();

private slots:
            void on_btnlogin_clicked();

    void openMainWindow();

    void on_btnsignup_clicked();

private:
    Ui::login *ui;
};

#endif // LOGIN_H
```

7.3 mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QListWidget>
#include <QMessageBox>
#include "Product.h"
#include "Order.h"
#include "Customer.h"
#include "orderwindow.h"
#include <vector>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow {
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

   void setCustomer(Customer* customer);

private slots:
    void onRemoveFromCart();
    void onApplyDiscount();
    void updateTotalLabel();
    void onProductSelected();

    void on_addButton_clicked();
    void onGiftWrappingToggled();

    void on_confirmOrder_clicked();

    double calculateTotalCost();

    void openOrderWindow(Order* order, Customer* customer);

private:
       Customer* customer; // Add this line
    Ui::MainWindow *ui;
    std::vector<Product> availableProducts; // Store available products
    std::vector<Product> cartItems;
    Order* currentOrder; // Pointer to the current order
    OrderWindow* orderWindow; // Pointer to the order window
    void populateProductList(); // Function to populate the product list
};
#endif // MAINWINDOW_H

```

7.4 order.h

```
#ifndef ORDER_H
```

```
#define ORDER_H

#include <QString>
#include <vector>
#include "Product.h"
#include "Shipment.h"

class Order {
private:
    int orderID;
    QString customerName;
    std::vector<Product> products;
    Shipment shipment;

public:
    Order(int id, const QString& name);

    int getOrderID() const;

    void confirmOrder();

    void selectPayment();

    void addProduct(const Product& product);

    void setShipmentMethod(const QString& method);

    double getTotalCost() const;

    void displayOrderDetails() const;

    void saveOrder() const;

    std::vector<Product> getProducts() const;
};

#endif // ORDER_H
```

7.5 orderwindow.h

```
#ifndef ORDERWINDOW_H
#define ORDERWINDOW_H

#include <QWidget>
#include "order.h"
#include "Customer.h"

class OrderWindow : public QWidget {
    Q_OBJECT

public:
    explicit OrderWindow(Order* order, Customer* customer, QWidget *parent = nullptr);

    ~OrderWindow();

private slots:
            void on_confirmButton_clicked();

    void on_paymentButton_clicked();

private:
    Ui::OrderWindow *ui;
    Order* order;
    Customer* customer;

    void displayOrderDetails();
};

#endif // ORDERWINDOW_H
```

7.6 Product.h

```
#ifndef PRODUCT_H
#define PRODUCT_H

#include <string>

class Product {
private:
    std::string name;
    double price;
    std::string description;
    int stock;
    std::string imagePath;

public:
    Product(const std::string& name, double price, const std::string& description, int stock, const
std::string& imagePath);

    std::string getName() const;

    double getPrice() const;

    std::string getDescription() const;

    int getStock() const;

    std::string getImagePath() const;

    void reduceStock(int quantity);

    void increaseStock(int quantity);
};

#endif // PRODUCT_H
```

7.7 Shipment.h

```
#ifndef SHIPMENT_H
#define SHIPMENT_H

#include <QObject>
#include <QString>

class Shipment : public QObject {
    Q_OBJECT

private:
    double shippingCost;
    QString shippingMethod;

public:
    explicit Shipment(QObject* parent = nullptr);

    void setShippingMethod(const QString& method);

    double getShippingCost() const;

    QString getMethod() const;
};

#endif // SHIPMENT_H
```

7.8 User.h

```
#ifndef USER_H
#define USER_H

#include <QString>

class User {
protected:
    int userID;
    QString username;
    QString password;

public:
    User(const QString& name, const QString& pwd);

    virtual ~User();

    bool verification(const QString& name, const QString& pwd, const QString& email) const;

    virtual void signup() = 0;

    virtual void login() = 0;

    QString getUsername() const;

    void setUsername(const QString& name);

    int getUserID() const;

    QString getPassword() const;

    void setPassword(const QString& pwd);

    bool doesExist(const QString& email) const;
};

#endif // USER_H
```