

SYD366NCC Individual Assignment - Theordinary.com/en-ca

Sahibpreet Singh - 165338211

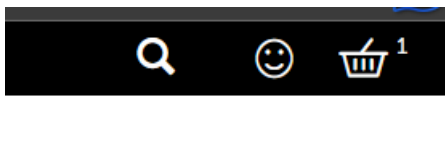
29 March 2023

Introduction

Theordinary.com is an e-commerce platform that caters to the needs of people for buying skincare products. The website offers products related to skin and provides a comprehensive user experience. This website was chosen for its easy navigation and clean UI. This report aims at providing an overview of the sign-up process and creating an order.

Sign-up Process

The sign-up process starts after clicking the smiley icon on the top right corner.



Once clicked, the user is redirected to a page where they can either sign in or create an account. To create an account, the user needs to provide their email address, first name, last name, phone number, and password. The user also has the option to sign up for The Ordinary's newsletter by checking the box provided.

Confirm Email

Password

Confirm Password

☐ KEEP ME UPDATED

*By checking the above box you are agreeing to receive email communications from DECIEM Inc., it affiliates, brands (The Ordinary and NIOD) and/or marketing partners. This can be changed at any time. Please refer to our [Privacy Policy](#) and [Terms of Use](#) for more details or [Contact Us](#).

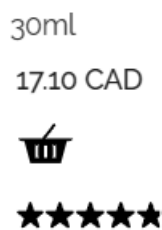
JOIN NOW

© 2020 DECIEM INC. All rights reserved. | [Privacy Policy](#) | [Terms of Use](#) | [Contact Us](#)

Once the user has filled out the required fields, they need to click on the 'Join Now' button to complete the sign-up process. After the process, the user is redirected to the home page.

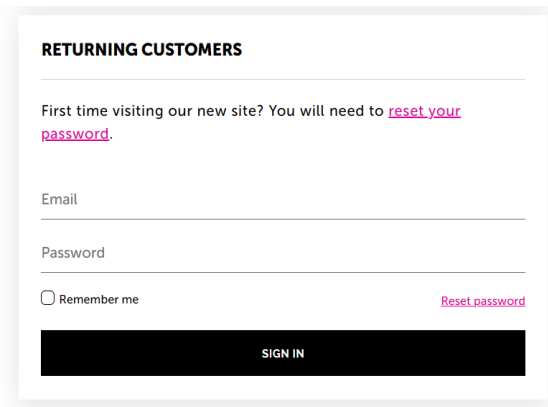
Create Order

Creating an order on the website is a simple process that can be completed in a few steps. The process starts by selecting the products the user wants to purchase by clicking on the product's image or clicking on the add to cart option.



The user can then continue adding products to the cart or proceed to checkout by clicking on the shopping cart icon located on the top right of the website beside the smiley icon displayed above. Once the user has reviewed their cart, they need to click on the 'Checkout' button to proceed with the purchase.

The user is then directed to a page where they can either sign in or check out as a guest. If the user has an account, they need to enter their email address and password to sign in. If the user does not have an account, they can either check out as a guest by providing their shipping and billing address, email address, and payment method or by creating an account. Once the user has filled out all the required fields, they need to click on the 'Continue to Payment' button to proceed with the purchase.



RETURNING CUSTOMERS

First time visiting our new site? You will need to [reset your password](#).

Email

Password

☐ Remember me [Reset password](#)

SIGN IN

The user is then directed to a page where they can select their payment method. Once the user has selected their preferred payment method, they need to fill out the required fields regarding the payment method chosen. Once the user has completed the payment, they need to click on the 'Complete Order' button to finalize the purchase. In the sequence diagram checkout as a guest isn't covered only the sign-in process has been covered for simplicity purposes.

Reflection

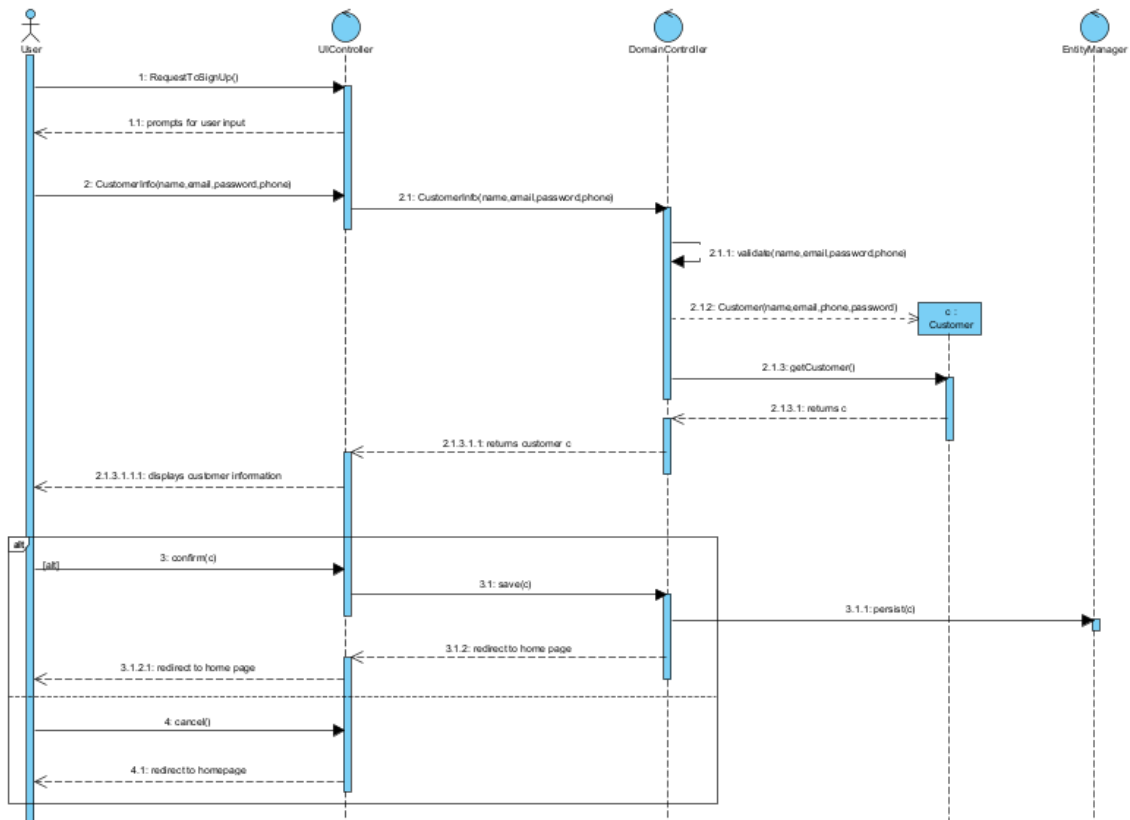
The website is well-developed with easy navigation and simple checkout and sign-up process. The amazing aspect is the UI which is far better than the other websites provided. The only thing I would like to change is it smiley icon for the user account. The reason is that it creates confusion for the users and results in them looking around for the sign-up or sign-in process. I would change it with a logo of a person to let the user know that its purpose is related to the user account.

```

classDiagram
    class Customer {
        -customerId: string
        -name: string
        -email: string
        -phone: string
        -username: string
        -password: string
        <<order>> Orders: Array<List<Order>>
        +getCustomerId(): string
        +setCustomerId(CustomerID): void
        +getName(): string
        +setName(name: string): void
        +getEmail(): string
        +setEmail(email: string): void
        +getPhone(): string
        +setPhone(phone: string): void
        +getUsername(): string
        +setUsername(username: string): void
        +getPassword(): string
        +setPassword(password: string): void
        +getOrders(): Array<List<Order>>
        +setOrders(Orders: Array<List<Order>>): void
    }
    class Order {
        -OrderID: string
        -Customer: Customer
        <<orderItems>> OrderItems: Array<List<OrderItem>>
        -SubTotal: double
        -Tax: double
        -Total: double
        +getOrderID(): string
        +setOrderID(OrderID: string): void
        +getCustomer(): Customer
        +setCustomer(Customer: Customer): void
        +getOrderItems(): Array<List<OrderItem>>
        +setOrderItems(OrderItems: Array<List<OrderItem>>): void
        +getSubTotal(): double
        +setSubTotal(SubTotal: double): void
        +getTax(): double
        +setTax(Tax: double): void
        +getTotal(): double
        +setTotal(Total: double): void
    }
    class OrderItem {
        -OrderItemID: string
        -product: Product
        -quantity: int
        -price: double
        <<product>> Product: Product
        +setProduct(product: Product): void
        +getQuantity(): int
        +setQuantity(quantity: int): void
        +getPrice(): double
        +setPrice(price: double): void
    }
    class Product {
        -ProductID: string
        -name: string
        -desc: string
        -price: double
        -instructions: string
        +getProductID(): string
        +setName(name: string): void
        +getDesc(): string
        +setDesc(desc: string): void
        +getPrice(): double
        +setPrice(price: double): void
        +getInstructions(): string
        +setInstructions(instructions: string): void
    }
    Customer "0..*" -- "1" Order : Places
    Order "1..*" -- "1" OrderItem : which contains
    OrderItem "1" -- "1" Product : for the
  
```

The diagram illustrates the relationships between four classes: Customer, Order, OrderItem, and Product. Customer (0..*) places an Order (1). An Order (1..*) contains one or more OrderItems (1). An OrderItem (1) is for a specific Product (1). The classes contain various attributes and methods for managing orders and products.

[illegible]



References

Theordinary.com/en-ca screenshots.