

Project Documentation

Project Name: EKart

Project Description: Ekart is a web-based e-commerce platform that allows users to buy products online. The platform features a user-friendly interface that makes it easy for customers to register (sign-up) themselves with the application, browse products, and add them to their cart. Then, they can place the order and do the payment either through a debit or credit card. It is built using modern web technologies and is designed to be highly scalable, secure, and reliable.

Major Modules:

- User Management
- Product Catalog
- Shopping Cart
- Payment

Technology Stack:

- **Backend:** Spring Boot, Spring Data, Spring REST
- **Frontend:** Angular, HTML5, CSS3, Bootstrap
- **Database:** MySQL

Non-Functional Requirements:

- Performance: The application should be fast and responsive, with quick load times and minimal lag or delay.
- Scalability: The application should be able to handle large amounts of traffic and scale as needed to accommodate growth.
- Security: The application should be secure and protect user data from unauthorized access, with measures such as encryption and secure authentication.
- Reliability: The application should be reliable and available, with minimal downtime or outages.
- Usability: The application should be easy to use and intuitive, with a clear and consistent user interface.
- Maintainability: The application should be easy to maintain and update, with clear and well-organized code that is easy to understand and modify.
- Compatibility: The application should be compatible with a wide range of devices and browsers, with support for different screen sizes and resolutions.

Final Deliverables:

- Complete source code
- Database DDL script
- Application archive (.jar) with source code
- Sample screenshots of important screens

User Stories:

SET 01: Product

US 01: View Products

As a Customer, I should be able to view the available products, so that I can buy the product I want.

Acceptance criteria:

- Only logged-in customers should be provided with this feature.
- The customer should be able to access the products from the home page.
- It should display all the available products with their names, brands, prices, and images.
- The customer should be able to click on a product to see more details, with "Add to Cart" button.

US 02: Search Product

As a Customer, I should be able to search for the product by its name or brand, so that I can buy that product.

Acceptance criteria:

- Only logged-in customers should be provided with this feature.
- The search bar should be visible and accessible in-home page of the application.
- The search bar should allow the customer to enter a search term, either a product name or brand name.

- The search should be case-insensitive and should return relevant results based on the search term.
- The search results page should display all relevant products with their names, brands, prices, and images.
- The customer should be able to click on a product to see more details, including category, and features.
- Display proper error messages in case the product is not available

SET 02: Cart

US 03: Add Item to Cart

As a customer, I should be able to add the items of my preference to my cart from the product details.

Acceptance criteria:

- Only logged-in customers should be provided with this feature.
- The customer should be able to access the product details by clicking on a product from the products page or from the search results page.
- The product details page should display all relevant information about the product, including its name, description, price, and image.
- The customer should be able to add the product to their cart by clicking on an "Add to Cart" button and the cart should update in real time.
- Verify if the same item is already present in the cart. We should get a message that the product is already added to the cart.
- Quantity should also be mentioned while adding items to the cart.
- The customer should be able to view the contents of their cart at any time by clicking on a "Cart" menu in the navigation bar.

- The system should save the customer's cart between sessions, allowing them to return to their cart and complete their purchase later if they choose.
- The customer should be able to proceed to checkout from their cart or continue shopping if they choose.
- Display proper messages in case of errors or exceptions

US 04: View Cart

As a customer, I should be able to view my cart, so that I can get to know the items that I am purchasing.

Acceptance criteria:

- Only logged-in customers should be provided with this feature.
- The customer should be able to access their cart from any page in the application by clicking on the "Cart" menu in the navigation bar.
- The cart should display all the items that the customer has added to it, along with their names, brands, prices, images, and quantities.
- For every product, a Remove button (Delete Icon) should be there to delete the product.
- There should be a "Place Order" button to proceed with buying the items in the cart.
- Display proper messages in case of the cart is empty

US 05: Update Items

As a Customer, I should be able to update the available number of products in my cart.

Acceptance criteria:

- Only logged-in customers should be provided with this feature.
- The customer should be able to see the quantity of each item in their cart and edit the quantity if necessary.
- The customer should be able to increase or decrease the quantity of each item in their cart using plus and minus buttons.
- A proper message should be displayed on updating the quantity.
- The cart should be updated in real-time to reflect any changes the customer makes to their cart.

US 06: Delete Items

As a registered customer, I should be able to delete the products in the cart so that items that are no longer required can be removed.

Acceptance criteria:

- Only logged-in customers should be provided with this feature.
- The customer is allowed to choose and delete the items from his cart.
- Customer should be prompted to confirm the delete operation before the delete operation is performed.
- On successful deletion, a successful message should be displayed to the customer.
- The cart should be updated in real-time to reflect any changes the customer makes to their cart.

- Display proper error message in case of any error or exception

SET 03: Order

US 07: Place Order

As a customer, I may want to place an order for the product(s) that are available in my cart.

Acceptance criteria:

- Only logged-in customers should be provided with this feature.
- The customer should be able to place the order from the cart.
- The updated total cost of all the items in the cart should be displayed.
- The customer should be able to use a secure payment method to complete the purchase.
- For making payment, a list of already added Debit/Credit cards should be displayed to choose from and there should be an option to add a new card as well.
- If payment is through a Credit card, a 10% discount should be applied else 5% discount should be applied
- Card details include the card type, card number, name on the card, and expiration date.
- On selecting the particular card, there should be a “Confirm Order” button
- On Clicking the Confirm Order button, an order id should be generated.
- The customer should be able to make a payment by entering the generated order id and CVV to complete the payment.
- A success message should be displayed after the payment is done.

- The inventory of the products should be updated after the customer successfully places an order.

US 08: Add New Card

As a customer, I want to add a new card, so that I can make payments easily while placing orders.

Acceptance criteria:

- Only logged-in customers should be provided with this feature.
- The customer should be able to add a new credit/debit card to their account by entering the card details which includes:
 - Card Type: Can be a Debit card or Credit card(should be selected from dropdown),
 - Card number: Should be 16 digits
 - CVV: Should be of 3 digits
 - Expiry Date: It should be a future date
 - Name on Card: Only alphabets and spaces are allowed, the length should not be more than 50 characters, and it should not contain only spaces.
 - Note: The CVV should be hashed before storing in the database.
- The application should validate the card information to ensure that it is accurate and complete.
- The application should securely store the credit/debit card information and follow all necessary security protocols for handling sensitive information.
- The customer should be able to select the newly added card also as the preferred payment method.

US 09: View Orders

As a customer, I should be able to view all the orders placed by me, so that I can see my orders.

Acceptance criteria:

- Only logged-in customers should be provided with this feature.
- The customer should be able to view a list of all their previous orders.
- Each order should display relevant information such as order id, date and time of purchase, delivery address, and payment information.
- The order history should be easily accessible from the customer's profile "My Orders" section.

API Documentation

API Name: CustomerAPI

Description: This API provides endpoints for customer registration, and login.

Base URL: /customer-api

Endpoints:

1. Login

POST: /login

Description: Authenticating the customer.

Request Body:

```
{  
  
  "emailId": "String",  
  
  "password": "String"  
}
```

Responses:

- 200 OK

```
{  
  
  "emailId": "String",  
  
  "name": "String",  
  
  "password": "String",  
  
  "newPassword": "String",
```

```
"phoneNumber": "String",  
  
"address": "String"  
  
}
```

- 400 Bad Request

No customer found with the email id / invalid data

2. Register

POST: /register

Description: Creating a new customer account.

Request Body:

```
{  
  
"emailId": "String",  
  
"name": "String",  
  
"password": "String",  
  
"newPassword": "String",  
  
"phoneNumber": "String",  
  
"address": "String"  
  
}
```

Responses:

- 200 OK

You are successfully registered as customer with Email Id: String

- 400 Bad Request

The email id is already in use. Please try with a new email id / invalid data

API Name: ProductAPI

Description: This API provides endpoints for fetching product details.

Base URL: /product-api

Endpoints:

1. Get All Products

GET: /products

Description: Fetching all the products.

Responses:

- 200 OK

```
[  
  {  
    "productId": Integer,  
    "name": "String",  
    "description": "String",  
    "category": "String",  
    "brand": "String",  
    "price": Double,  
    "availableQuantity": Integer  
  }  
]
```

2. Get Specific Product

GET: /product/{productId}

Description: Fetching the product with given Id.

Path Variable: productId

Responses:

- 200 OK

```
{  
  
  "productId": Integer,  
  
  "name": "String",  
  
  "description": "String",  
  
  "category": "String",  
  
  "brand": "String",  
  
  "price": Double,  
  
  "availableQuantity": Integer  
  
}
```

- 400 Bad Request

Sorry product is not available

API Name: CartAPI

Description: This API provides endpoints for fetching, adding and updating the products of the cart.

Base URL: /cart-api

Endpoints:

1. Add Product to Cart

POST: /products

Description: Adding the product to customer's cart.

Request Body:

```
{  
  "customerEmailId": "String",  
  "cartProducts":  
    [  
      {  
        "product":  
          {  
            "productId": Integer  
          },  
        "quantity": Integer  
      }  
    ]  
}
```

Responses:

- 200 OK

The products are successfully added to the cart having cartId : Integer

- 400 Bad Request

Invalid data

2. Get Products from Cart

GET: /customer/{customerEmailId}/products

Description: Fetching the products from customer's cart.

Path Variable: customerEmailId

Responses:

- 200 OK

```
[  
  {  
    "cartProductId": Integer,  
    "product": {  
      "productId": Integer,  
      "name": "String",  
      "description": "String",  
      "category": "String",  
      "brand": "String",  
      "price": Double,  
      "availableQuantity": Integer
```

```
    },  
    "quantity": Integer  
  }  
]
```

- 400 Bad Request

Sorry No cart found for you / No products are available in your cart

3. Update Quantity of Product

PUT: /customer/{customerEmailId}/product/{productid}

Description: Updating the quantity of products in customer's cart.

Path Variables: customerEmailId, productid

Request Body:

Integer

Responses:

- 200 OK

Quantity updated successfully

- 400 Bad Request

Product is not available in your cart / Sorry No cart found for you /
invalid data

4. Delete Product from Cart

DELETE: /customer/{customerEmailId}/product/{productid}

Description: Deleting the product from customer's cart.

Path Variables: customerEmailId, productid

Responses:

- 200 OK
Your item has been removed from cart
- 400 Bad Request
 - Product is not available in your cart / Sorry No cart found for you / invalid data

API Name: OrderAPI

Description: This API provides endpoints for placing order and fetching order details.

Base URL: /order-api

Endpoints:

1. Place Order

POST: /place-order

Description: Placing the order of products in customer's cart.

Request Body:

```
{  
  
  "customerEmailId": "String",  
  
  "dateOfDelivery" : "LocalDateTime",  
  
  "paymentThrough" : "String"  
}
```

Responses:

- 200 OK

Order is successfully placed with order id : Integer

- 400 Bad Request

Invalid data

2. Get Orders

GET: /customer/{customerEmailId}/orders

Description: Fetching the order details of customer.

Path Variable: customerEmailId

Responses:

- 200 OK

```
[  
  {  
    "orderId": Integer,  
    "customerEmailId": "String",  
    "dateOfOrder": "LocalDateTime",  
    "totalPrice": Double,  
    "orderStatus": "String",  
    "discount": Double,  
    "paymentThrough": "String",  
    "dateOfDelivery": "LocalDateTime",  
    "deliveryAddress": "String",  
    "orderedProducts": [  
      {
```

```

        "orderedProductId": Integer,

        "product": {

            "productId": Integer,

            "name": "String",

            "description": "String",

            "category": "String",

            "brand": "String",

            "price": Double,

            "availableQuantity": Integer

        },

        "quantity": Integer

    }

]

}

]

```

- 400 Bad Request

No orders are found for customer email id

API Name: PaymentAPI

Description: This API provides endpoints for managing cards (debit/credit) and making payment.

Base URL: /payment-api

Endpoints:

1. Add Card

POST: /customer/{customerEmailId:.+}/cards

Description: Adding the card for payment of customer's cart.

Path Variable: customerEmailId

Request Body:

```
{  
  
  "cardType": "String",  
  
  "cardNumber": "String",  
  
  "nameOnCard": "String",  
  
  "cvv": Integer,  
  
  "expiryDate": "LocalDate",  
  
  "customerEmailId": "String"  
  
}
```

Responses:

- 200 OK
The card has been successfully added, with card ID: Integer
- 400 Bad Request
No customer found with the email id / invalid data

2. Get Cards

GET: /customer/{customerEmailId}/card-type/{cardType}

Description: Fetching the card details of customer.

Path Variable: customerEmailId, cardType

Responses:

- 200 OK

```
[  
  {  
    "cardType": "String",  
    "cardNumber": "String",  
    "nameOnCard": "String",  
    "hashCvv": "XXX",  
    "cvv": null,  
    "expiryDate": "LocalDate",  
    "cardId": Integer,  
    "customerEmailId": "String"  
  }  
]
```

- 400 Bad Request

No card found

3. Make Payment

POST: /customer/{customerEmailId}/order/{orderId}

Description: Making payment for order of customer products.

Path Variable: customerEmailId, orderId

Request Body:

```
{  
  
  "cardId" : Integer,  
  
  "cvv" : Integer  
}
```

Responses:

- 200 OK
We have received the payment of – Rs. Double for Order id Integer
- 400 Bad Request
No customer found with the email id / Order not found / The transaction already executed for given order id / Are you sure you want to pay for order which doesn't belongs to you? / invalid data

Frontend Documentation

The Ekart frontend is the user-interface part of the e-commerce platform where customers browse products, place orders, and manage their accounts. It is typically a web application that runs in a browser and communicates with the Ekart backend through APIs. It is built using web technologies such as HTML, CSS, Bootstrap, Angular framework to enable modular and scalable development.

Project Setup:

- Download the project from given link.
- Install Node.js and Angular CLI if not already installed.
- Run **npm install** to install project dependencies.
- Run **ng serve** to start the application.
- Navigate to <http://localhost:4200/> to view the app in the browser.

Folder Structure:

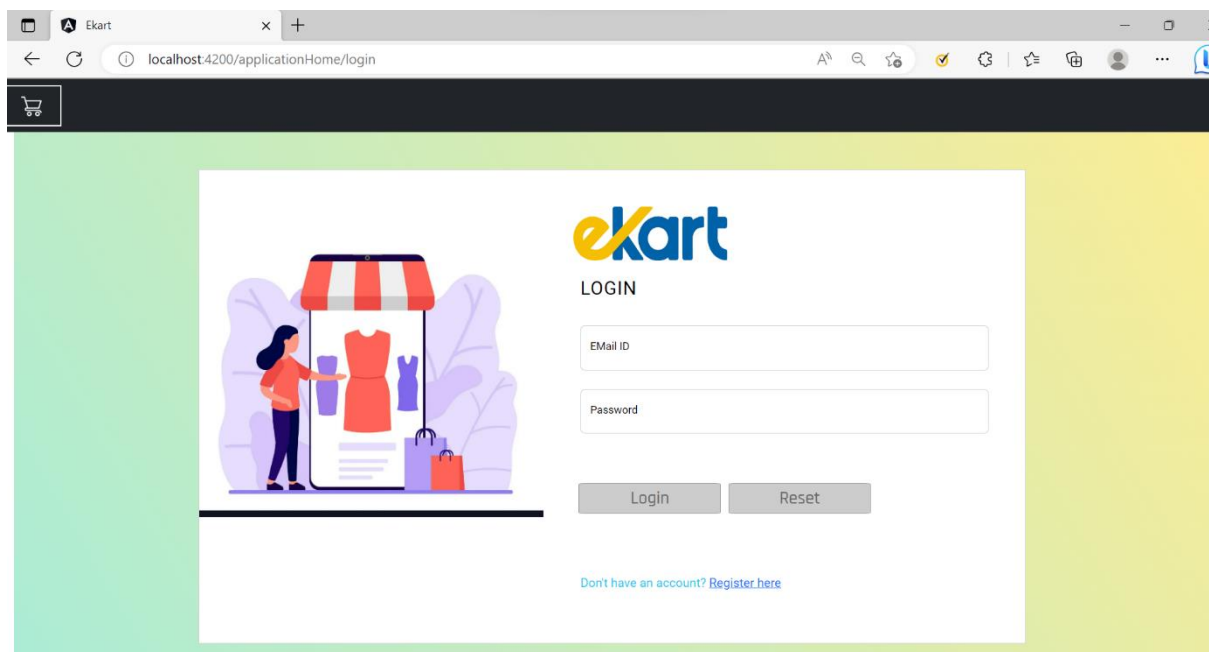
- **src** directory contains the source code for the application, including the main application module, and other resources.
- **app** directory contains the main application module and components, including the root component.

- **assets** directory contains static assets used by the application, such as images.
- **environments** directory contains environment-specific configuration files, such as 'environment.ts' for development.
- **index.html** is the main HTML file for the application, which is served by the browser and contains the base HTML structure.
- **style.css** is the main CSS file for the application, which contains the global styles used by the application.
- **main.ts** is the main TypeScript file for the application, which bootstraps the main application module ('AppModule').
- **tsconfig.json** is the configuration file for the TypeScript compiler, which specifies the TypeScript version, and other settings.
- **angular.json** is the configuration file for the Angular CLI, which specifies the project settings.
- **node_modules** directory contains the dependencies installed by npm, which are required to run the application.
- **package.json** is the configuration file for the npm package manager, which specifies the project dependencies, scripts, and other metadata.

Components

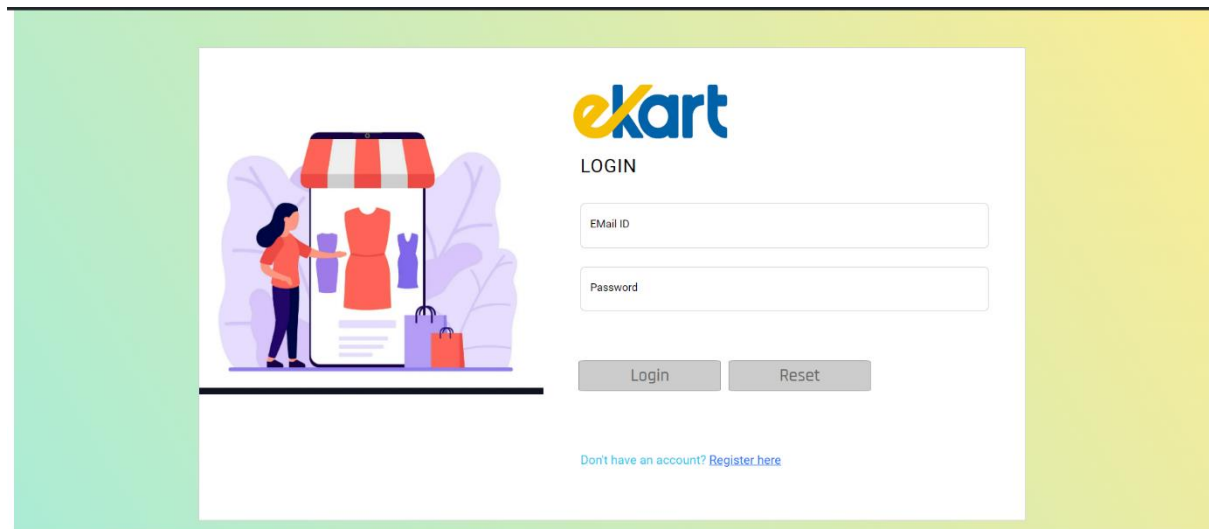
- **customer-landing-page component:**

This component includes a navigation bar with a link that redirects a customer to the landing page. Please find below sample screenshot for your reference:

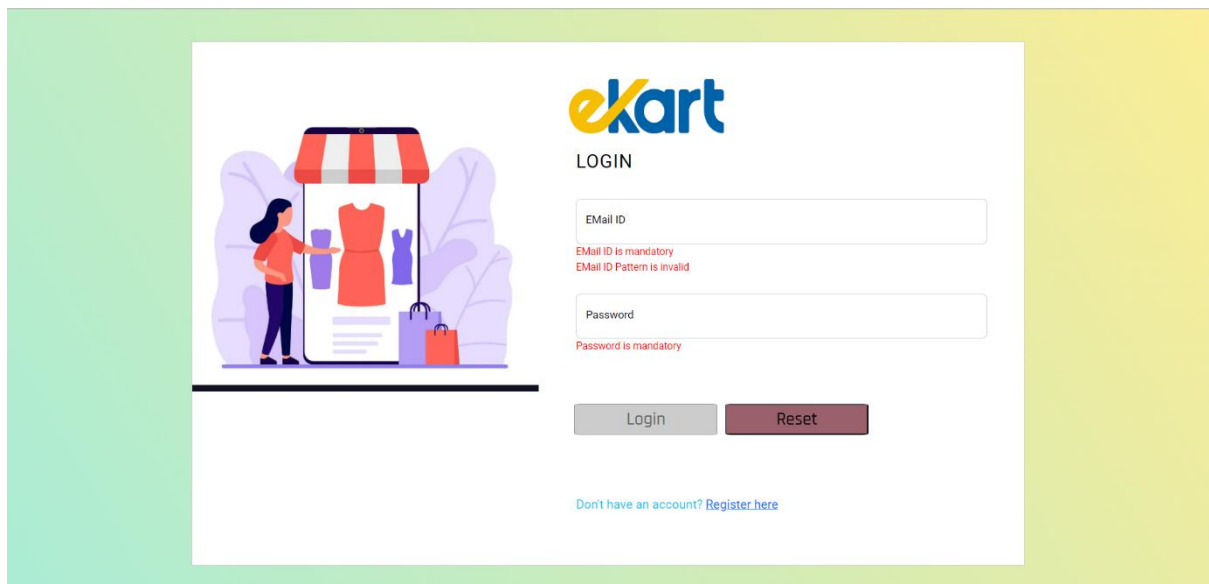


- **login component:**

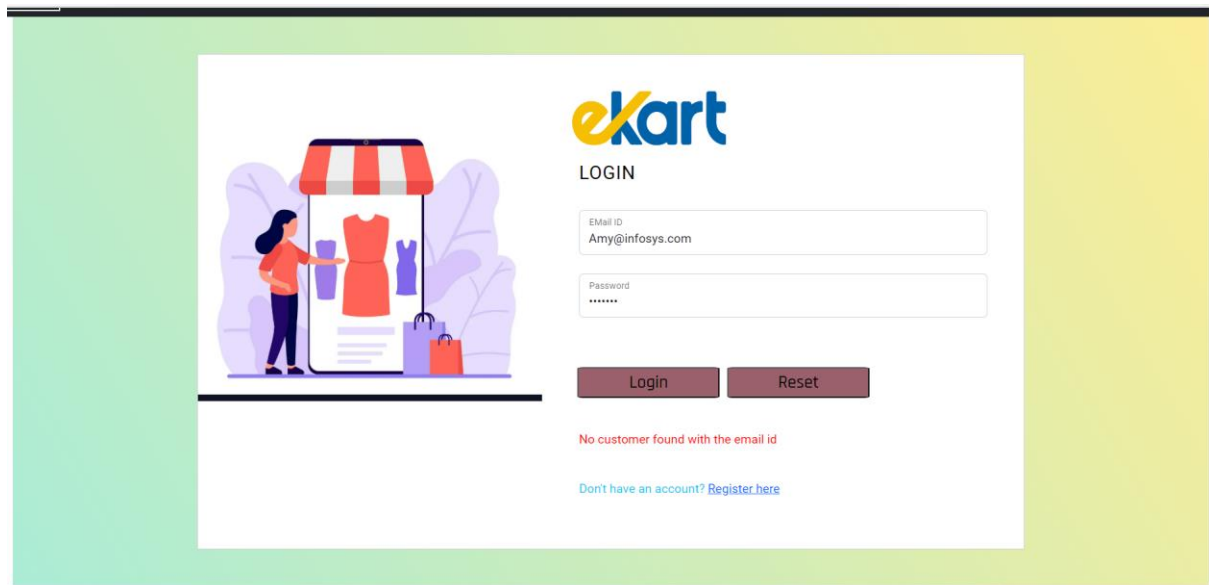
The login component is used to allow users to log in to an existing account, using their email and password. 'Register here' link is also present on login page for new users to create a new account. It look like:



For invalid data, appropriate validation message should be displayed.



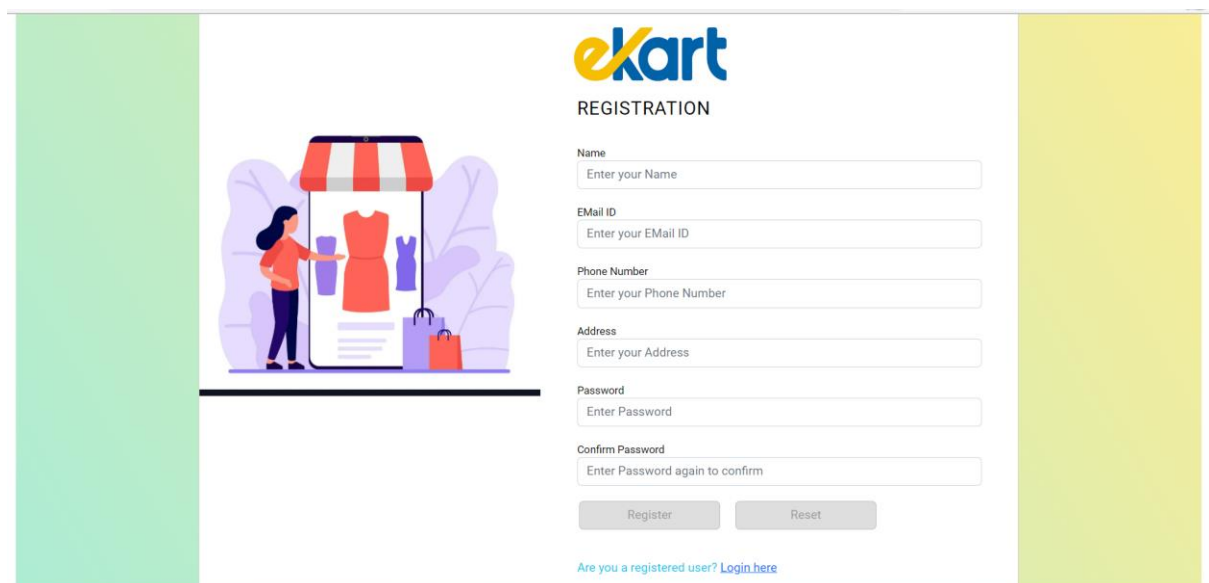
After form submission, appropriate success or error message should come.



The image shows a web browser window displaying the 'ekart' login page. On the left, there is an illustration of a woman in a red shirt and black pants standing next to a clothing rack with a red and white striped awning. The rack contains a red dress, a blue dress, and a purple bag. To the right of the illustration is the 'ekart' logo in blue and yellow. Below the logo, the word 'LOGIN' is displayed in black. There are two input fields: 'Email ID' with the text 'Amy@infosys.com' and 'Password' with a masked password '*****'. Below these fields are two buttons: 'Login' and 'Reset'. A red error message 'No customer found with the email id' is displayed below the buttons. At the bottom, there is a link: 'Don't have an account? [Register here](#)'.

- **registration component:**


This register component is used to allow users to create an account so that they can log in to an application. It should prompt the users to enter their details such as name, email, phone number and address. Users should also set the password for their account.



The image shows a web browser window displaying the 'ekart' registration page. On the left, there is an illustration of a woman in a red shirt and black pants standing next to a clothing rack with a red and white striped awning. The rack contains a red dress, a blue dress, and a purple bag. To the right of the illustration is the 'ekart' logo in blue and yellow. Below the logo, the word 'REGISTRATION' is displayed in black. There are six input fields: 'Name' (placeholder: 'Enter your Name'), 'Email ID' (placeholder: 'Enter your EMail ID'), 'Phone Number' (placeholder: 'Enter your Phone Number'), 'Address' (placeholder: 'Enter your Address'), 'Password' (placeholder: 'Enter Password'), and 'Confirm Password' (placeholder: 'Enter Password again to confirm'). Below these fields are two buttons: 'Register' and 'Reset'. At the bottom, there is a link: 'Are you a registered user? [Login here](#)'.

The details entered by users should be validated as per the validations given below:

- Name should contain only alphabets with a single space between the words.
- Email ID should be a valid email address with domains.
- Phone Number should be of 10 digits.
- Password should contain at least one uppercase, one lowercase, one digital and one special character.



Name

Enter your Name

Name is mandatory
Name Pattern is invalid

Email ID

Enter your EMail ID

Email ID is mandatory
Email ID Pattern is invalid

Phone Number

Enter your Phone Number

Phone Number is mandatory
Phone Number Pattern is invalid

Address

Enter your Address

Address is mandatory

Password

Enter Password

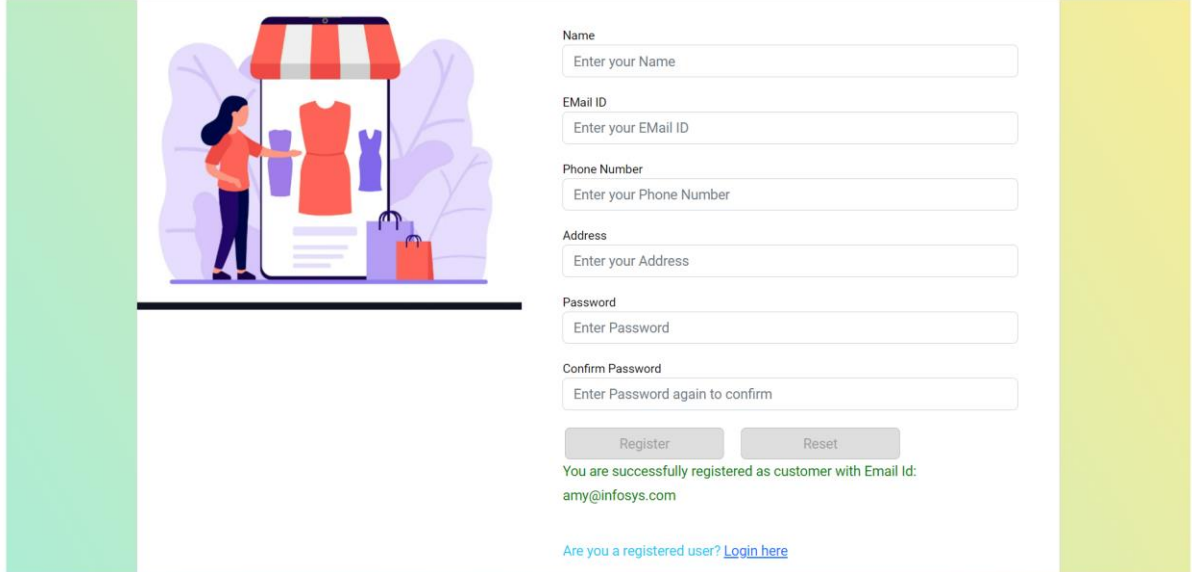
Password is mandatory
Password Pattern is invalid

Confirm Password

Enter Password again to confirm

Please re-enter the password

Sample Registration may look like:



The registration form is set against a light green background. On the left, there is an illustration of a woman in a red top and dark pants standing next to a clothing rack with a red and white striped awning. The rack holds a red dress, a blue dress, and a purple top. A red shopping bag sits on the floor next to the rack. To the right of the illustration is a registration form with the following fields: Name (with placeholder 'Enter your Name'), EMail ID (with placeholder 'Enter your EMail ID'), Phone Number (with placeholder 'Enter your Phone Number'), Address (with placeholder 'Enter your Address'), Password (with placeholder 'Enter Password'), and Confirm Password (with placeholder 'Enter Password again to confirm'). Below these fields are two buttons: 'Register' and 'Reset'. A green message states: 'You are successfully registered as customer with Email Id: amy@infosys.com'. At the bottom, there is a link: 'Are you a registered user? [Login here](#)'.

Name
Enter your Name

EMail ID
Enter your EMail ID

Phone Number
Enter your Phone Number

Address
Enter your Address

Password
Enter Password

Confirm Password
Enter Password again to confirm

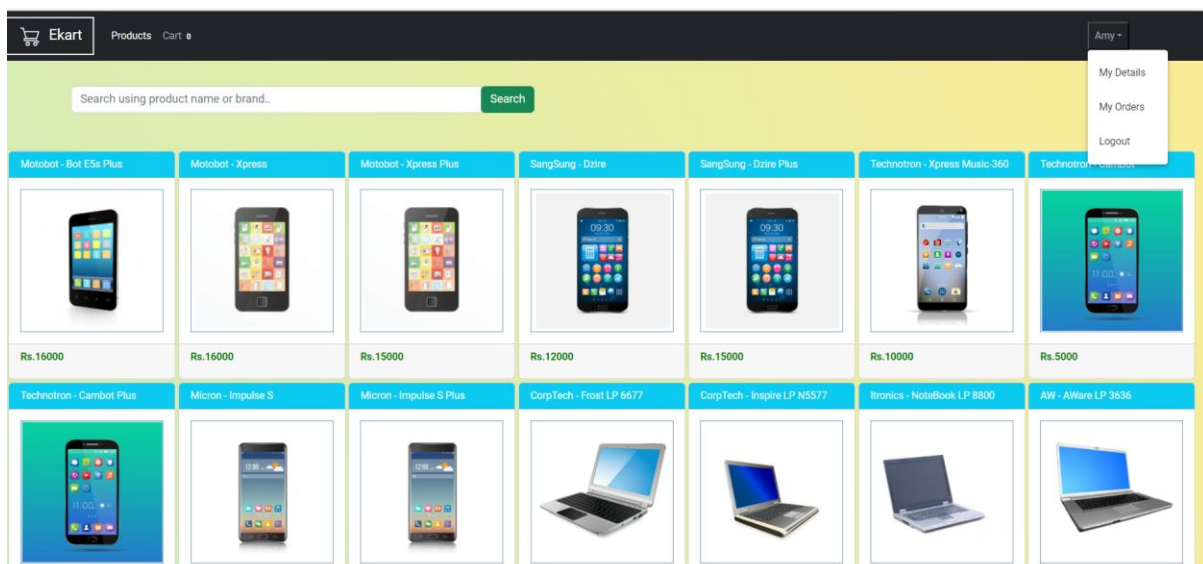
Register Reset

You are successfully registered as customer with Email Id:
amy@infosys.com

Are you a registered user? [Login here](#)

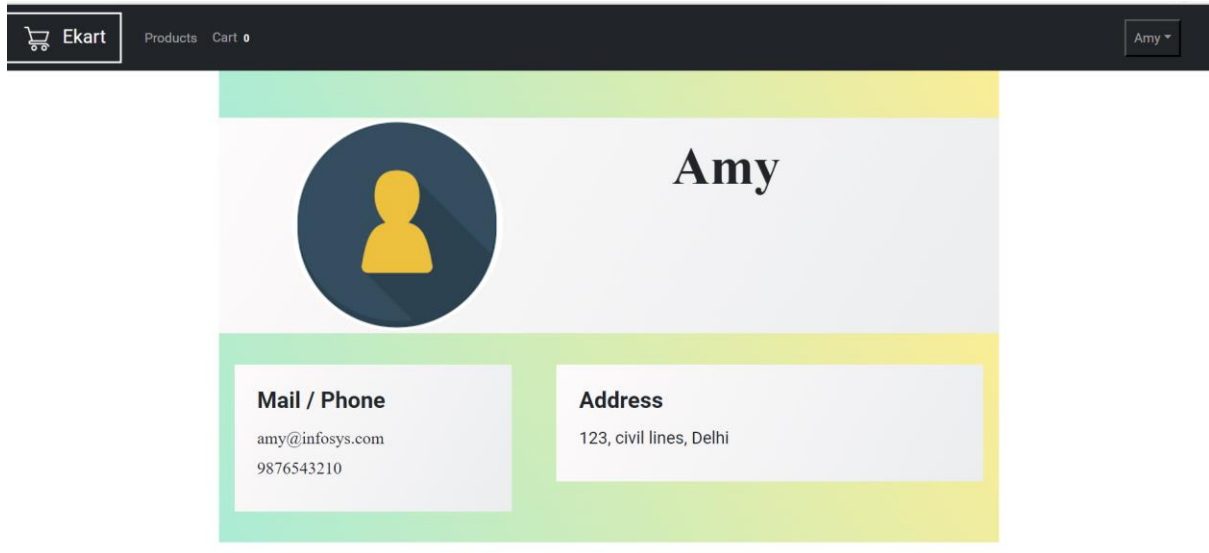
- **customer-home component:**

This component includes a navigation bar with links (for Cart, Products and Ekart landing page) and a profile drop down of logged-in customer. Please find below sample screenshot for your reference:



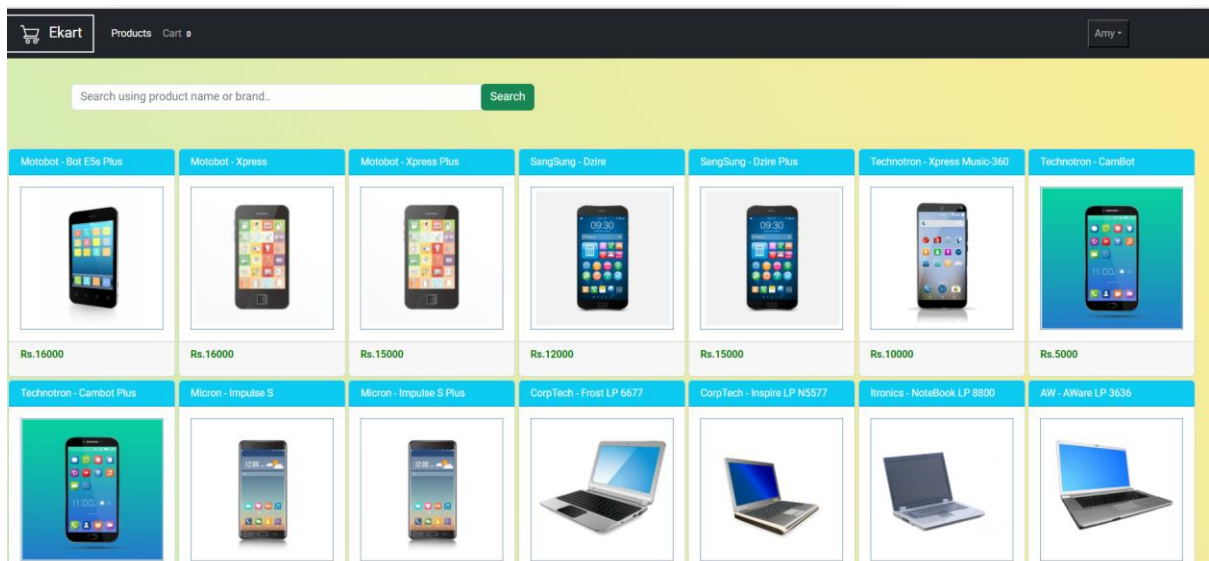
- **customer-details component:**

This user profile component is used to display the user's personal information including their name, email, address, and contact information. 'My Details' should be accessible from navigation menu tab.

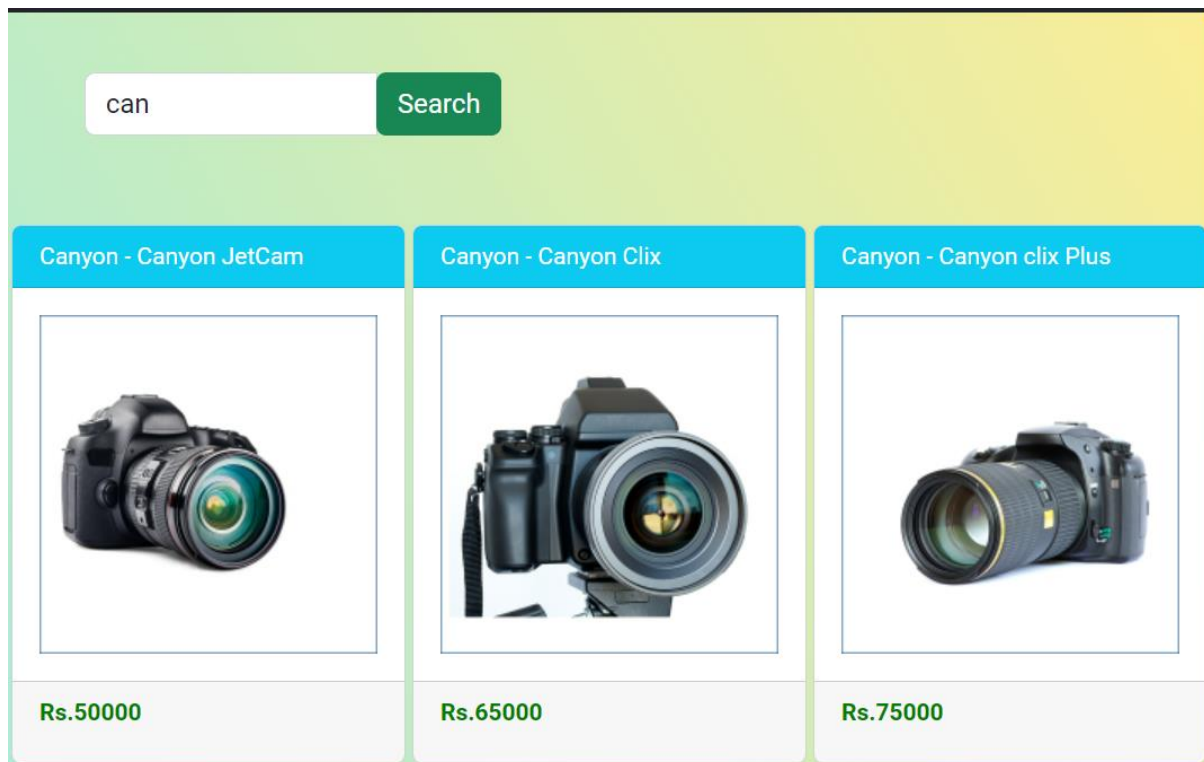


- **view-all-products component:**

This product list component is used to display a list of products including their name, brand, price, and image.



It also consists of a search bar using which a customer can search for specific product based on name or brand. When the user start typing in the search bar then product names or brands containing the typed characters should appear as suggestions.

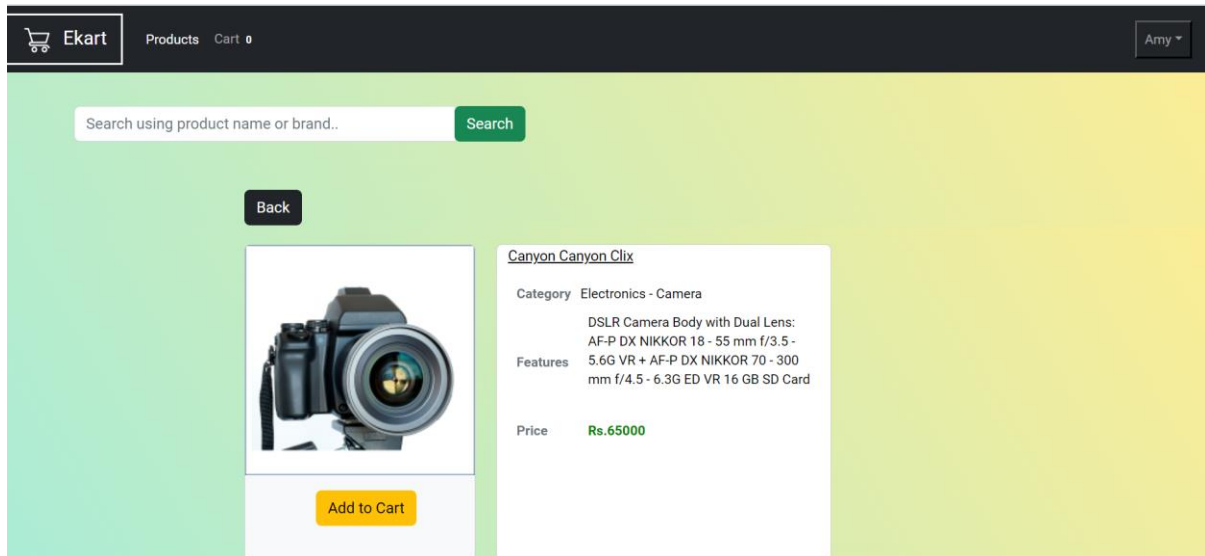


If products are not available, then corresponding message will be displayed.

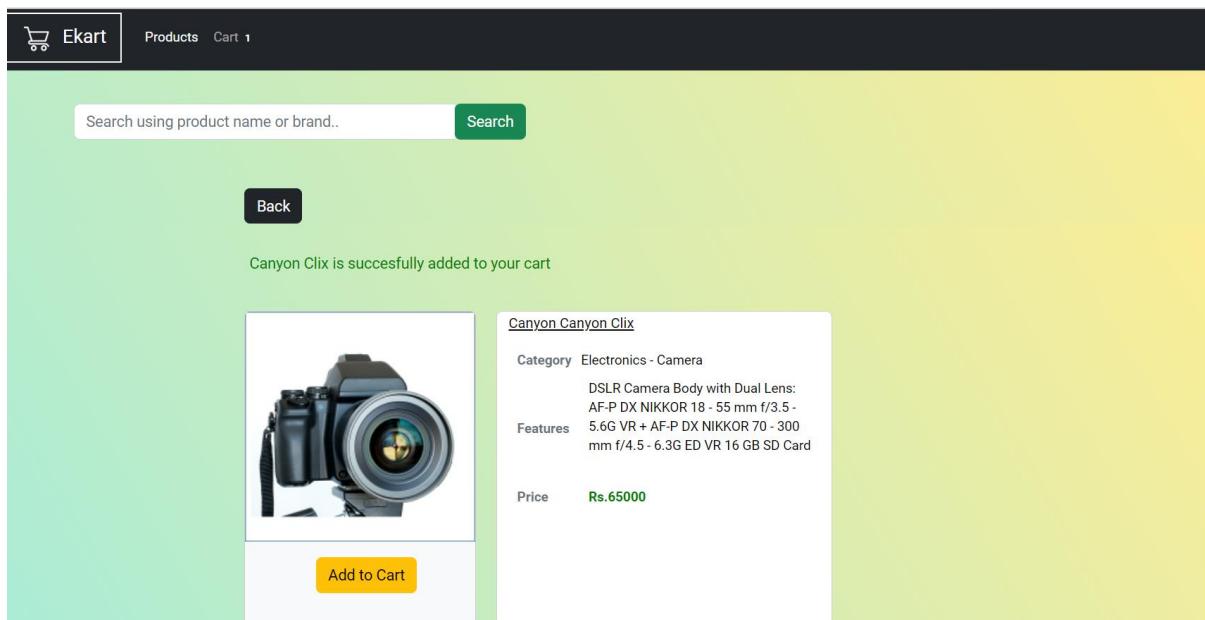
No products available!!

- **product-details component:**

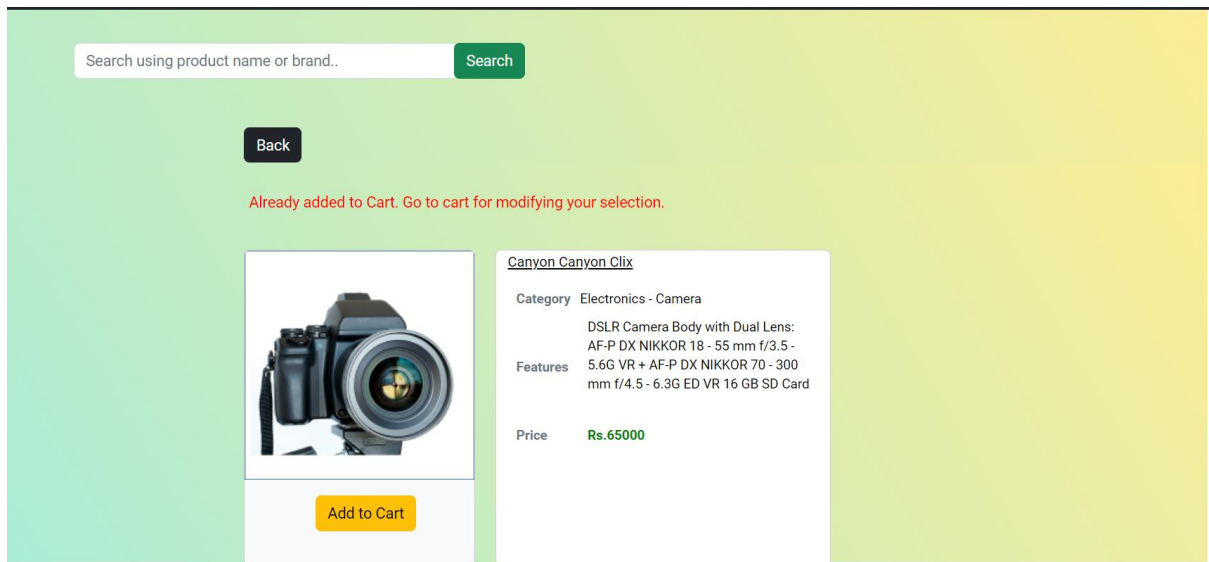
The product details component is used to display the details of a specific product, including its name, brand, description, price, and image as:



On clicking “Add to Cart” button, the product will be added to the cart and message will be displayed as:

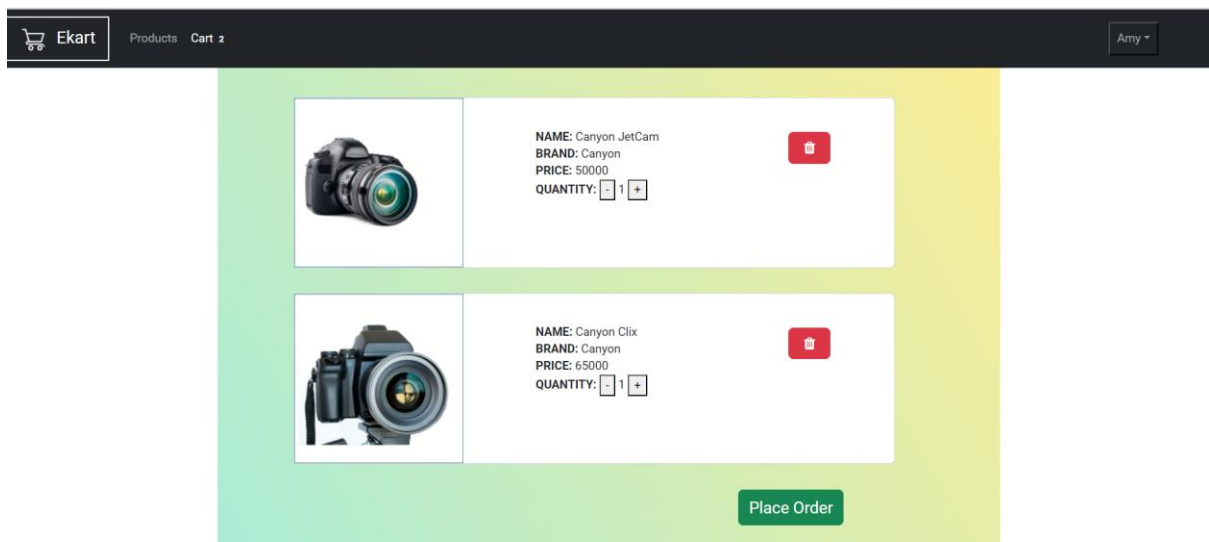


Customers should not be able to add a product again which is already present in their cart.

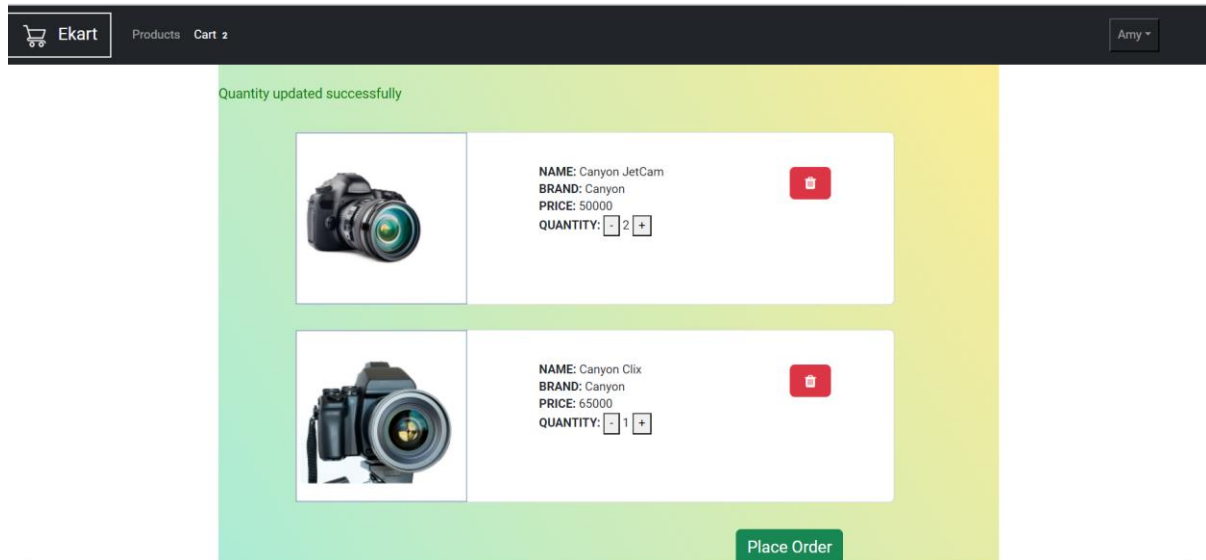


- **customer-cart component:**

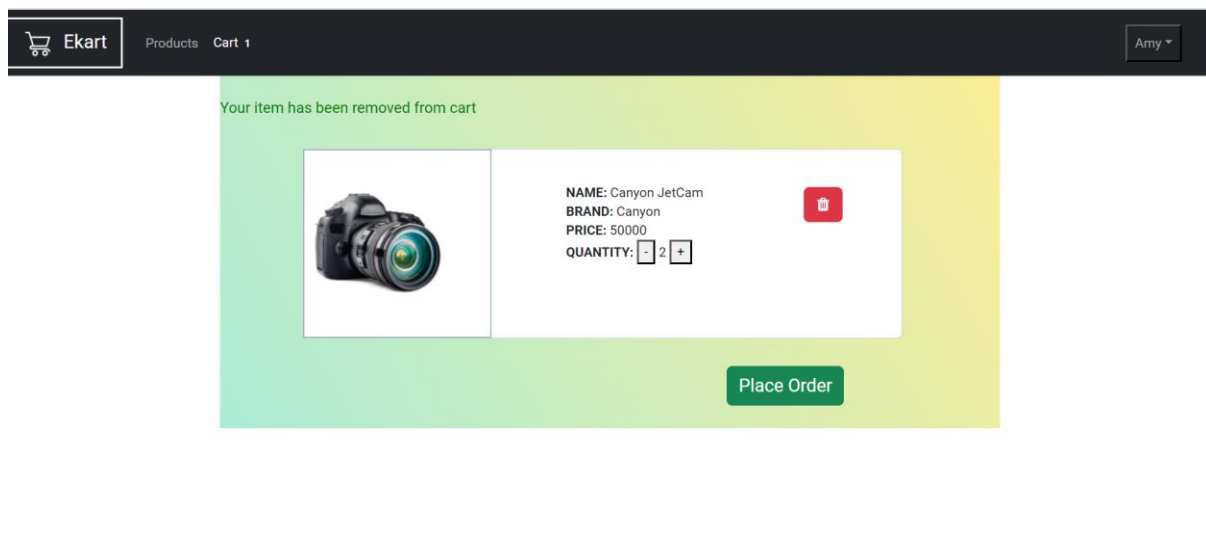
This cart component is used to display the items added to the user's shopping cart, including their name, brand, price, and quantity, and to allow the user to update or remove items from the cart.

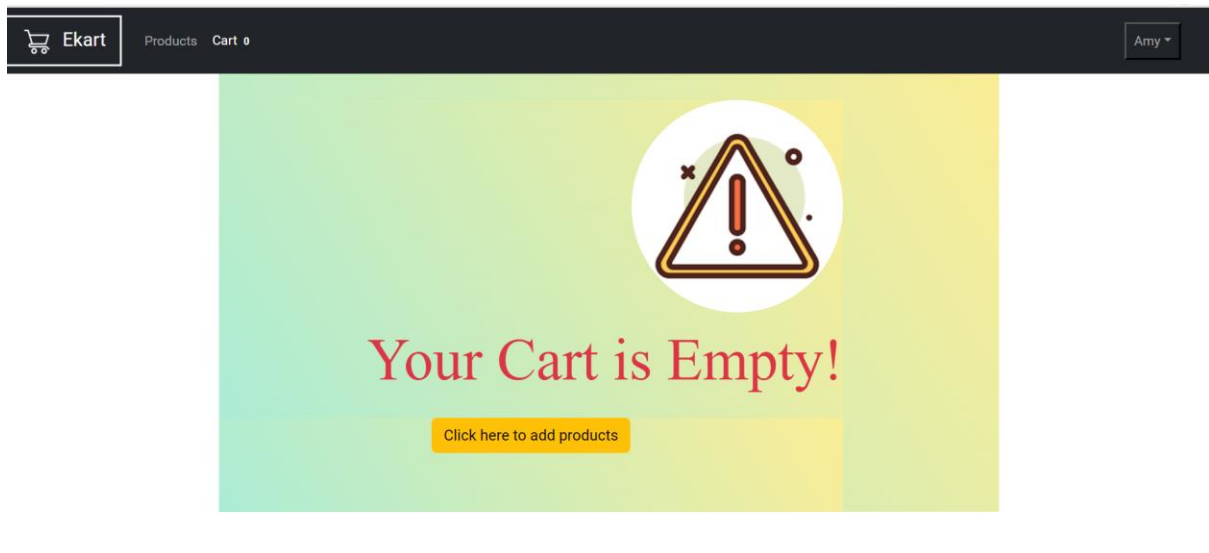


‘+’ and ‘-’ buttons are used to update the quantity of products in the cart. On updating the quantity, subtotal should also be updated and a proper message should be displayed as:



Delete icon is used to delete the product from cart. On deleting the product, proper message should be displayed as:



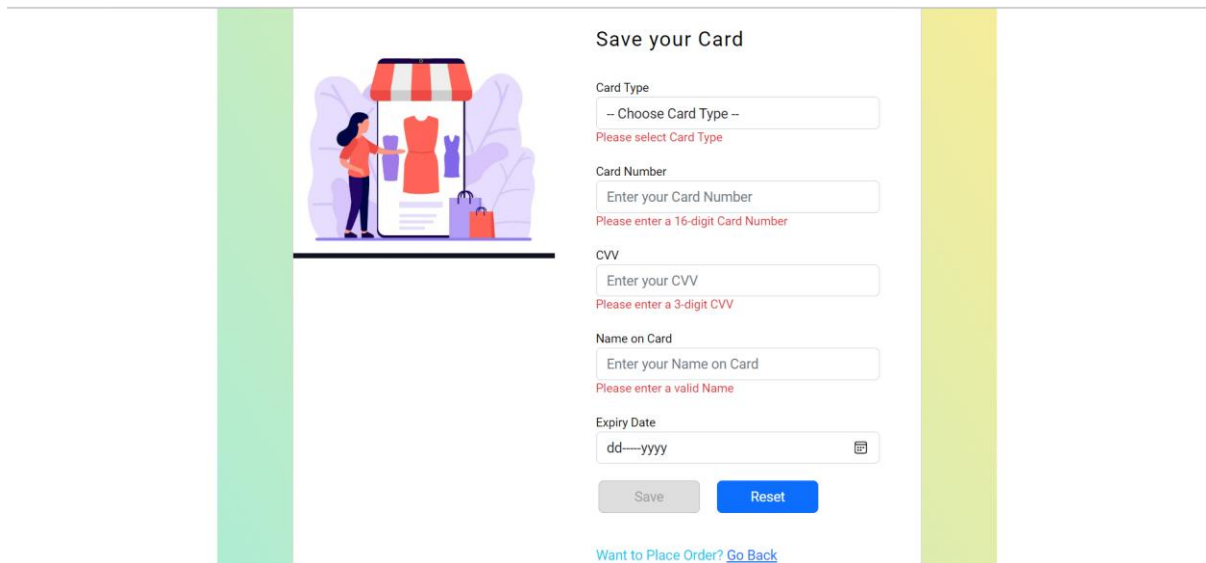


- **customer-card component:**

It is used to allow customers to add new credit or debit cards to their accounts for use during secure payment and manage transactions easily. The customer must provide the details such as Card Type, Card Number, CVV, Name on Card, and Expiry Date. Only valid details should be accepted:

- Card Type should be either Credit or Debit.
- Card Number should be of 16 digits.
- CVV should be of 3 digits.
- Name on Card should contain only English letters with single space between words.
- Expiry Date should be of future date.

The card form should be displayed as:

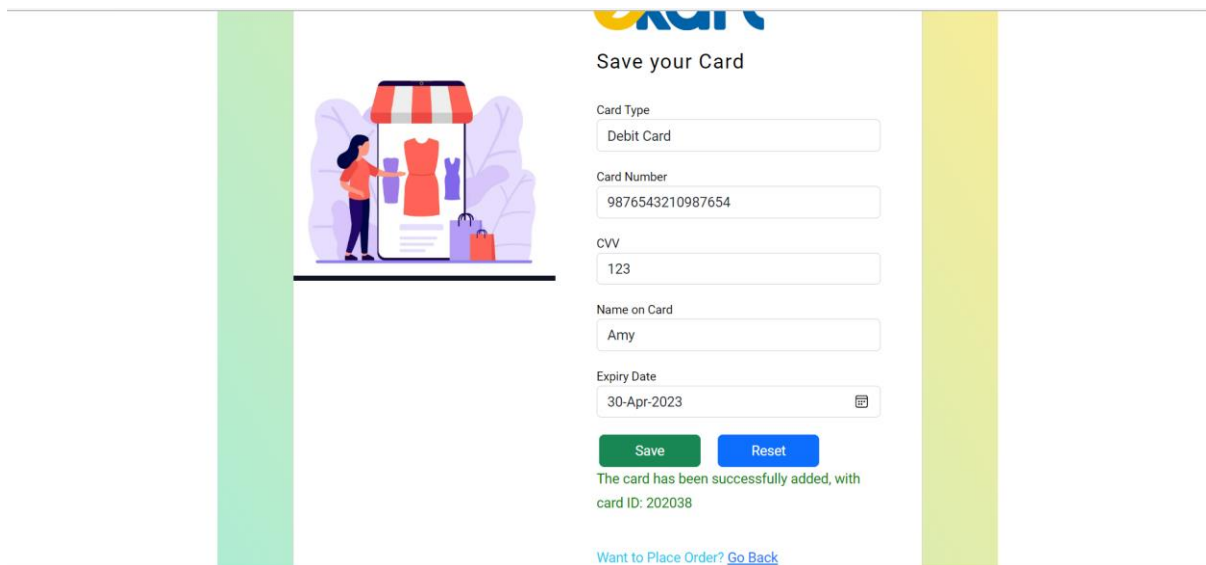


The screenshot shows a web interface for saving a credit card. On the left is a shopping cart illustration with a woman, a red dress, and a shopping bag. The main form is titled "Save your Card" and contains the following fields:

- Card Type:** A dropdown menu with the placeholder text "-- Choose Card Type --". Below it is a red error message: "Please select Card Type".
- Card Number:** A text input field with the placeholder text "Enter your Card Number". Below it is a red error message: "Please enter a 16-digit Card Number".
- CVV:** A text input field with the placeholder text "Enter your CVV". Below it is a red error message: "Please enter a 3-digit CVV".
- Name on Card:** A text input field with the placeholder text "Enter your Name on Card". Below it is a red error message: "Please enter a valid Name".
- Expiry Date:** A date picker field with the placeholder text "dd-----yyyy".

At the bottom of the form are two buttons: "Save" (grey) and "Reset" (blue). Below the buttons is a link: "Want to Place Order? [Go Back](#)".

Sample card adding may look like:



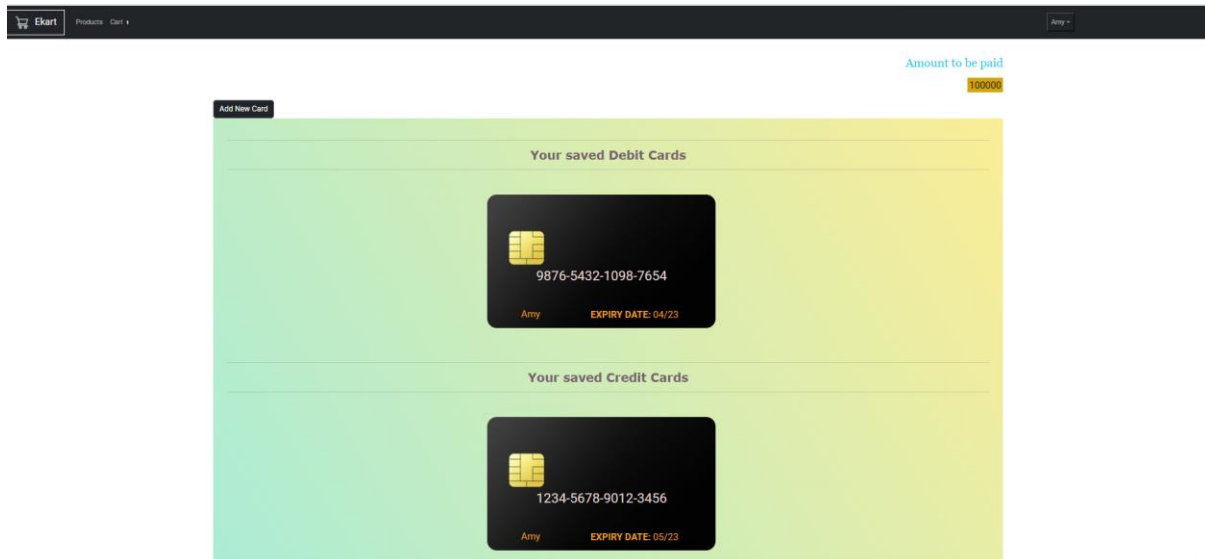
This screenshot shows the same "Save your Card" form, but with sample data entered and a success message. The "OKAY" logo is visible at the top left. The form fields are filled as follows:

- Card Type:** "Debit Card"
- Card Number:** "9876543210987654"
- CVV:** "123"
- Name on Card:** "Amy"
- Expiry Date:** "30-Apr-2023"

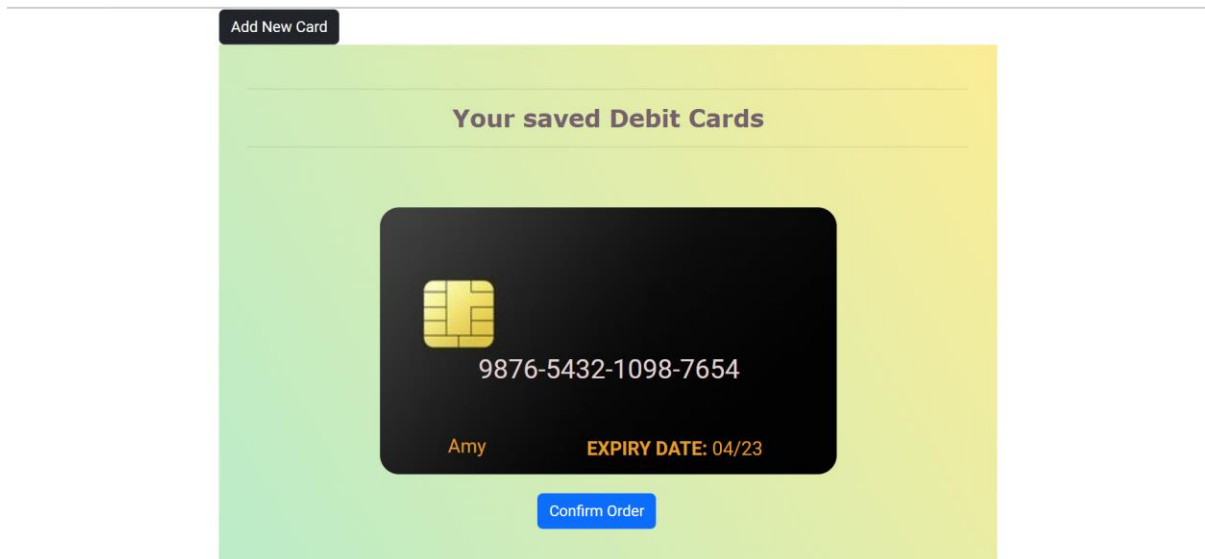
The "Save" button is now green, and the "Reset" button is blue. Below the buttons, a green success message reads: "The card has been successfully added, with card ID: 202038". The link "Want to Place Order? [Go Back](#)" remains at the bottom.

- **place-order component:**

It is used to allow customers to place an order for products they have added to their shopping cart and finalize their purchase. It displays the total amount to be paid and the card details.



On selecting the card, it will ask for a confirmation.



On confirming the order, ID will be generated for that order.

Order is successfully placed with order id 90007

Amount to be paid 100000

Add New Card

Your saved Debit Cards

9876-5432-1098-7654

ARMY EXPIRY DATE: 04/23

Confirm Order

Enter Order ID:
Enter the OrderID generated above

Enter CVV:
Enter your Debit Card CVV

Pay

Also, it will ask for CVV of the selected card for payment.

Amount to be paid 100000

Add New Card

Your saved Debit Cards

9876-5432-1098-7654

ARMY EXPIRY DATE: 04/23

Confirm Order

Enter Order ID:
90007

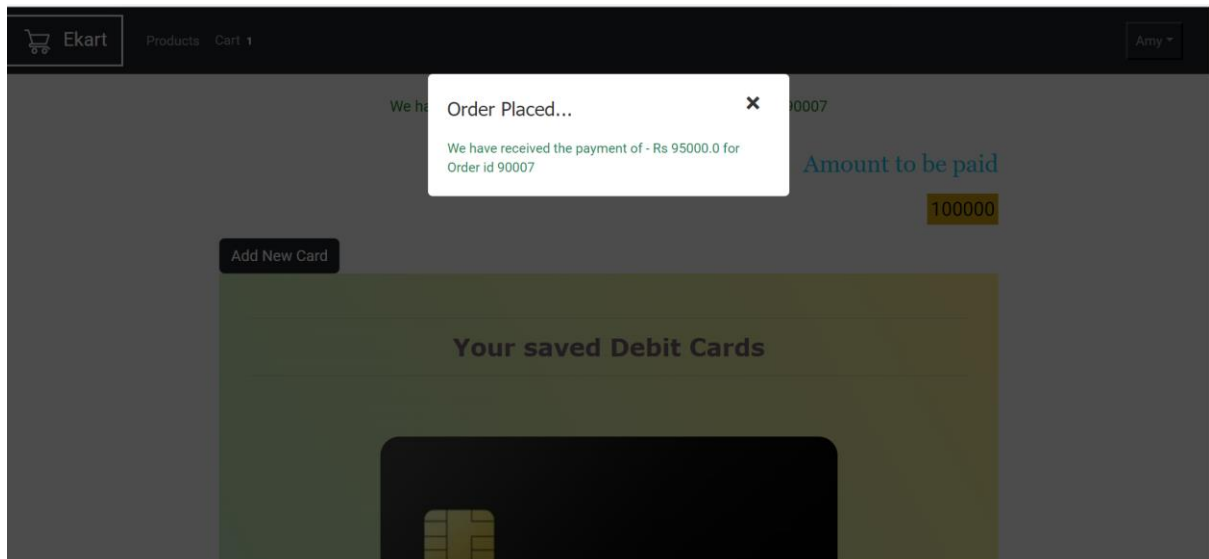
Enter CVV:

CVV should be of 3 digit

Pay

Your saved Credit Cards

On providing valid data, payment will happen. A discount of 10% if credit card and 5% if debit card will be applied.



- **view-order component:**

This component is used to display the user's order summary, including order number, date & time of order, shipping address, payment method and final price.

