TRENDY GIFT SHOP

**Requirements Analysis:**

Trendy Gift Shop sells three types of gift item: Greeting Cards, Souvenirs, and Toys.

Every item has unique ID, item description, the price, and available quantity in the store stored in the system.

The system can create a report to show the inventory of the available items, the description of each item, and the amount of sales per day for an item at any point.

The visitor must register with the website for the online purchase. The visitor must provide personal details including an email-id, username and password.

After the visitor login the system, it can add the available items to the shopping cart. As the items are selected and added to the shopping cart, website tracks the total amount to be paid for the items in shopping cart.

Bill is issued to the customer after the customer checkout. After the payment has been completed using the valid credit card, the checkout is completed.

Every details of the successful transaction are stored in the customers’ account

**Classes:**

Following classes are used to develop this system:

* GiftItem
  + Attributes
    - id,
    - price,
    - quantity,
    - description
  + Methods
    - get\_ID, set\_ID,
    - get\_price, set\_price
    - get\_quantity, set\_quantity
    - get\_description, set\_description
* GreetingCard (child off GiftItem)
  + Attributes
    - size
  + Methods
    - get\_size, set\_size
* Souvenir (child off GiftItem)
  + Attributes
    - souvenir\_type
  + Methods
    - get\_souvenir\_type, set\_souvenir\_type
* Toy (child off GiftItem)
  + Attributes
    - toy\_type
  + Methods
    - get\_toy\_type, set\_toy\_type
* Customer
  + Attributes
    - user\_ID
    - email
    - password
    - username
    - password
    - cart
    - order\_history
    - card
  + Methods
    - sign\_up
    - login
    - get\_user\_ID, set\_user\_ID
    - get\_email, set\_email
    - get\_password, set\_password
    - get\_username, set\_username
    - get\_order\_history
* Cart
  + Attributes
    - GiftItems
  + Methods
    - addItem
    - checkout
* Card
  + Attributes
    - card\_number
  + Methods
    - charge
    - get\_card\_details
    - set\_card\_details
* Website
  + Attributes
    - customers
    - gift\_items
  + Methods
    - register
    - add\_gift\_item
    - get\_customers, get\_items

**Relationships:**

Greeting Card, Toy, Souvenir are extended from the GiftItem class.

Website is composed of gift items.

Website is associated with the customer.

Customer is associated with the card and cart.

Cart is associated with the GiftItem.

**UML Diagram**:

A close up of a map

Description automatically generated

**Flow chart:**

**A close up of a map

Description automatically generated**

**Code**

#######

# GIFT ITEMS

#######

class GiftItem:

    def \_\_init\_\_(self, item\_id, price, quantity, description):

        self.\_id = item\_id

        self.\_price = price

        self.\_quantity\_available = quantity

        self.\_description = description

    def get\_id(self):

        return self.\_id

    def set\_id(self, item\_id):

        self.\_id = item\_id

    def get\_description(self):

        return self.\_description

    def set\_description(self, description):

        self.\_description = description

    def get\_price(self):

        return self.\_price

    def set\_price(self, price):

        self.\_price = price

    def set\_quantity\_available(self, quantity):

        self.\_quantity\_available = quantity

    def get\_quantity\_available(self):

        return self.\_quantity\_available

class GreetingCard(GiftItem):

    def \_\_init\_\_(self, item\_id, price, quantity, description, size):

        super().\_\_init\_\_(id, price, quantity, description)

        self.\_\_size = size

    def get\_size(self):

        return self.\_\_size

    def set\_size(self, size):

        self.\_\_size = size

class Souvenir(GiftItem):

    def \_\_init\_\_(self, item\_id, price, quantity, description, souvenir\_type):

        super().\_\_init\_\_(item\_id, price, quantity, description)

        self.\_\_souvenir\_type = souvenir\_type

    def get\_souvenir\_type(self):

        return self.\_\_souvenir\_type

    def set\_souvenir\_type(self, souvenir\_type):

        self.souvenir\_type = souvenir\_type

class Toy(GiftItem):

    def \_\_init\_\_(self, item\_id, price, quantity, description, toy\_type):

        super().\_\_init\_\_(item\_id, price, quantity, description)

        self.\_\_toy\_type = toy\_type

    def get\_toy\_type(self):

        return self.\_\_toy\_type

    def set\_toy\_type(self, toy\_type):

        self.\_\_toy\_type = toy\_type

########

# CART

########

class Cart:

    def \_\_init\_\_(self):

        self.\_\_GiftItems = []

    def addItem(self, gift\_item):

        self.\_\_GiftItems.append(gift\_item)

    def checkout(self, card):

        total\_price = 0

        for gift\_item in self.\_\_GiftItems:

            print(gift\_item.get\_ID, gift\_item.get\_price())

            total\_price += gift\_item.get\_price()

        self.\_\_GiftItems = []

        return card.charge(total\_price)

#########

#CARD

#########

class Card:

    def \_\_init\_\_(self, cardnumber):

        self.\_\_card\_number = cardnumber

    def charge(self, amount):

        print(amount, " charged on card: ", self.\_\_card\_number)

    def get\_card\_details(self):

        return self.\_\_card\_number

    def set\_card\_details(self, cardnumber):

        self.\_\_card\_number = cardnumber

#########

# CUSTOMER

########

class Customer:

    def \_\_init\_\_(self):

        self.\_\_user\_ID = None

        self.\_\_email = ""

        self.\_\_cart = Cart()

        self.\_\_order\_history = []

        self.\_\_card = None

    def sign\_up(self):

        self.\_\_user\_ID = int(input('Enter User ID: '))

        self.\_\_email = input('Enter Email: ')

        self.\_\_username = input('Enter User Name: ')

        self.\_\_password = input('Enter password: ')

        self.\_\_card = Card(input('Enter card: '))

    def login(self, username, password):

        if self.\_\_password == password and self.\_\_username == username:

            return True

        else:

            False

    def get\_user\_ID(self):

        return self.\_\_userID

    def set\_user\_ID(self, userid):

        self.\_\_userID = userid

    def get\_email(self):

        return self.\_\_email

    def set\_email(self, email):

        self.\_\_email = email

    def get\_username(self):

        return self.\_\_username

    def set\_username(self, uname):

        self.\_\_username = uname

    def get\_paswword(self):

        return self.\_\_password

    def set\_paswword(self, password):

        self.\_\_password = password

    def checkout(self):

        self.\_\_cart.checkout(self.\_\_card)

        self.\_\_order\_history.append(Cart)

    def get\_order\_history(self):

        for order in self.\_\_order\_history:

            print(order)

###########

# WEBSITE

##########

class Website:

    def \_\_init\_\_(self):

        self.\_\_customers = []

        self.\_\_gift\_items = []

    def register(self, customer):

        customer.sign\_up()

        self.\_\_customers.append(customer)

    def add\_gift\_item(self, gift\_item):

        self.\_\_gift\_items.append(gift\_item)

    def get\_customers(self):

        for customer in self.\_\_customers:

            print(customer.get\_username())

    def get\_items(self):

        for item in self.\_\_gift\_items:

            print(item.get\_description())

            print(item.get\_quantity\_available())

**Conclusion:**

By implementing a system of gift shop, we practically learnt the concepts of OOP for example inheritance, composition, accessibility of members.

Using python for the as a object-oriented language was a challenge as there is no keyword for the public, private and protected.

This system can be further extended by adding further details and complex functionalities.