Cairo University

Faculty of Computers and Artificial Intelligence

CS251

Introduction to Software Engineering

Toffee's SDS

Software Design Specifications

Draft Version

ID	Name	Email
20210501	Laila Hesham Kandil	lailakandil64@gmail.com
20211034	Doaa Ali El-Sayed	doali246885@gmail.com
20210458	Youssef Ahmed Zakaria	youssefahmed052@gmail.com

Software Design Specification

Contents

CS251: Phase 2 – <Team Name>
Project: Toffee

Software Design Specification

Team

ID	Name	Email	Mobile
20210501	Laila Hesham Kandil	lailakandil64@gmail.com	01012960328
20211034	Doaa Ali El-Sayed	doali246885@gmail.com	01019271286
20210458	Youssef Ahmed Zakaria	youssefahmed052@gmail.com	01120700202

Document Purpose and Audience

Purpose:

The purpose of an SDS document for the project called Toffee, which is an e-commerce platform, is to provide a comprehensive overview of the system's design and functionality. The system architecture will depict how Toffee's e-commerce platform works and what its parts are, such as the user interface, payment gateway, order processing system, and database. It will help people understand how Toffee's e-commerce platform behaves and what it can do. The architecture will cover various aspects of Toffee's e-commerce platform, like how it interacts with users, manages orders, and ensures data privacy and security.

The sequence diagram for Toffee's e-commerce platform will illustrate the interactions between its components during a particular sequence of events, such as a user searching for a product, adding it to the cart, and placing an order. It will show how different components of Toffee's e-commerce platform, such as the search engine, product catalog, cart management, and payment gateway, interact with each other to fulfill the user's request.

The state diagram for Toffee's e-commerce platform will show the different states that the system can be in, such as idle, searching, cart management, and order processing, and how it transitions between those states based on user actions and system events.

Together, these diagrams will provide a clear representation of Toffee's e-commerce platform's design and behavior, which can be used by developers, testers, and other stakeholders to understand and evaluate its functionality and make informed decisions about its development and deployment.

Software Design Specification

Audience:

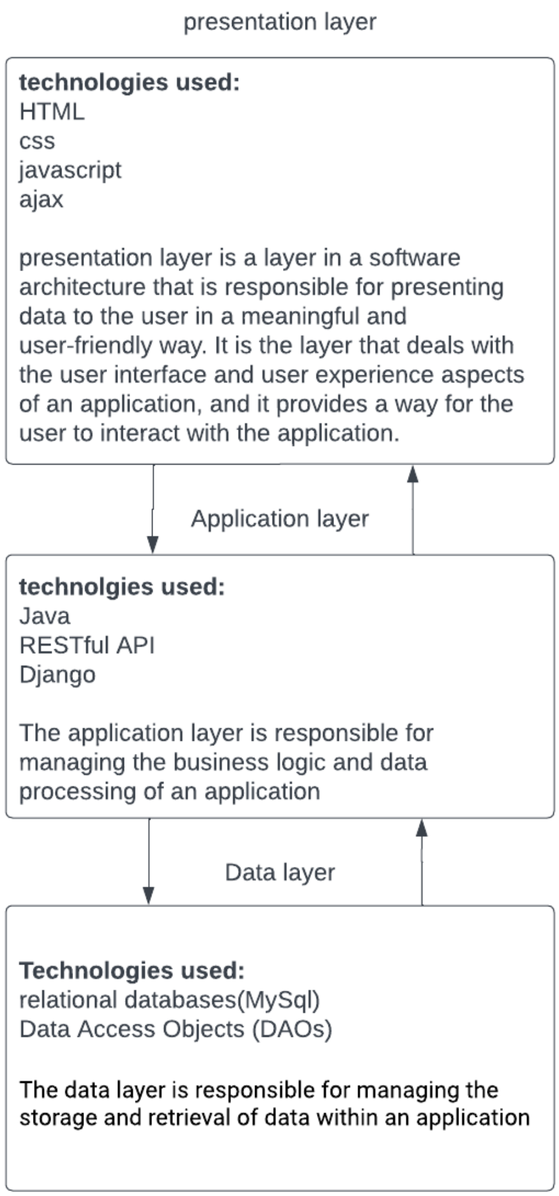
The audience of 3-tier architecture includes people who are involved in the design, development, and deployment of software systems.

- Software developers.
- Software architects.
- Project managers.
- Technical decision-makers.

Software Design Specification

System Models

I. Architecture Diagram



presentation layer

technologies used:
HTML
css
javascript
ajax

presentation layer is a layer in a software architecture that is responsible for presenting data to the user in a meaningful and user-friendly way. It is the layer that deals with the user interface and user experience aspects of an application, and it provides a way for the user to interact with the application.

Application layer

technolgies used:
Java
RESTful API
Django

The application layer is responsible for managing the business logic and data processing of an application

Data layer

Technologies used:
relational databases(MySql)
Data Access Objects (DAOs)

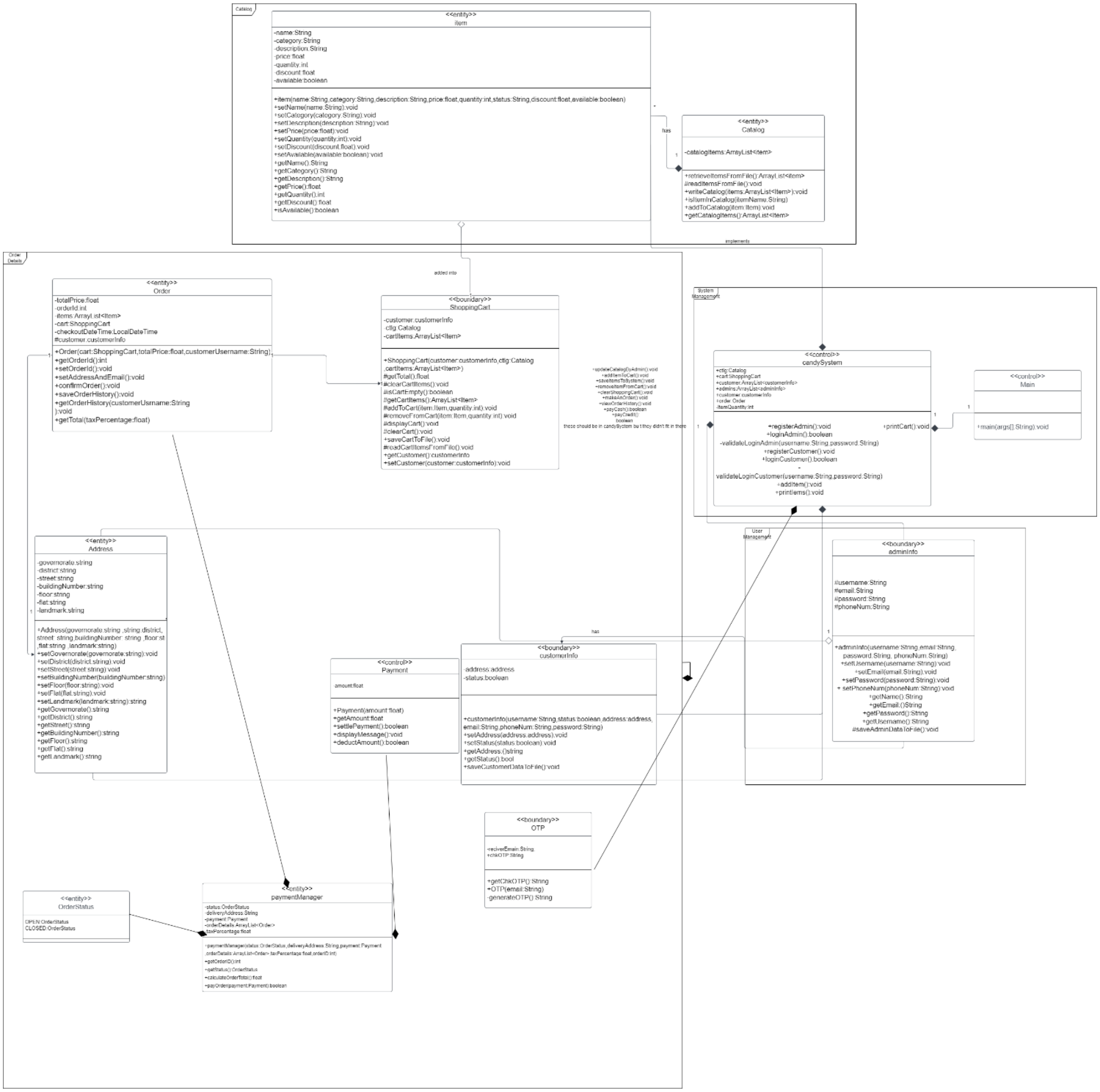
The data layer is responsible for managing the storage and retrieval of data within an application

Advantages of Three-Tier Architecture :

- The key three-tier benefit is improved scalability since the application servers can be deployed on many machines. Also, the database does not make longer connections with every client – it only requires connections from a smaller number of application servers.
- It improves data integrity. Here, all the updated information goes through the second tier. The second tier can ensure that only important information is allowed to be updated in the database and the risk of unreliable client applications corrupting information is removed.
- Security is improved since the client does not have direct access to the database; it is more difficult for a client to obtain unauthorized data. Business logic is more secure because it is stored on a secure central server.
- High performance, lightweight, persistent object.
- Better to reuse.
- Easy to maintain and modify.
- The added modularity makes it easier to modify or replace one tier without affecting the other tier.

Software Design Specification

II. Class Diagram(s)


Catalog
<<entity>>
item
-name:String
-category:String
-description:String
-price:float
-quantity:int
-discount:float
-available:boolean
+item(name:String,category:String,description:String,price:float,quantity:int,status:String,discount:float,available:boolean)
+setName(name:String):void
+setCategory(category:String):void
+setDescription(description:String):void
+setPrice(price:float):void
+setQuantity(quantity:int):void
+setDiscount(discount:float):void
+setAvailable(available:boolean):void
+getName():String
+getCategory():String
+getDescription():String
+getPrice():float
+getQuantity():int
+getDiscount():float
+isAvailable():boolean
*
has
1
<<entity>>
Catalog
-catalogItems:ArrayList<item>
+retrieveItemsFromFile():ArrayList<item>
#readItemsFromFile():void
+writeCatalog(items:ArrayList<Item>):void
+isItemInCatalog(itemName:String)
+addToCatalog(item:Item):void
+getCatalogItems():ArrayList<Item>
implements
added into
Order Details
<<entity>>
Order
-totalPrice:float
-orderId:int
-items:ArrayList<Item>
-cart:ShoppingCart
-checkoutDateTime:LocalDateTime
#customer:customerInfo
+Order(cart:ShoppingCart,totalPrice:float,customerUsername:String)
+getOrderId():int
+setOrderId():void
+setAddressAndEmail():void
+confirmOrder():void
+saveOrderHistory():void
+getOrderHistory(customerUsrname:String
):void
+getTotal(taxPercentage:float)
1
1
1
<<boundary>>
ShoppingCart
-customer:customerInfo
-ctlg:Catalog
-cartItems:ArrayList<Item>
+ShoppingCart(customer:customerInfo,ctlg:Catalog
,cartItems:ArrayList<Item>)
#getTotal():float
#clearCartItems():void
#isCartEmpty():boolean
#getCartItems():ArrayList<Item>
#addToCart(item:Item,quantity:int):void
#removeFromCart(item:Item,quantity:int):void
#displayCart():void
#clearCart():void
+saveCartToFile():void
#readCartItemsFromFile():void
+getCustomer():customerInfo
+setCustomer(customer:customerInfo):void
+updateCatalogByAdmin():void
+addItemToCart():void
+saveItemsToSystem():void
+removeItemFromCart():void
+clearShoppingCart():void
+makeAnOrder():void
+viewOrderHistory():void
+payCash():boolean
+payCredit():
boolean
these should be in candySystem bu t they didn't fit in there
System Management
<<control>>
candySystem
+ctlg:Catalog
+cart:ShoppingCart
+customer:ArrayList<customerInfo>
+admins:ArrayList<adminInfo>
+customer:customerInfo
+order:Order
-itemQuantity:int
+registerAdmin():void
+loginAdmin():boolean
-validateLoginAdmin(username:String,password:String)
+registerCustomer():void
+loginCustomer():boolean
-
validateLoginCustomer(username:String,password:String)
+addItem():void
+printIems():void
+printCart():void
1
1
1
<<control>>
Main
+main(args[]:String):void
User Management
<<boundary>>
adminInfo
#username:String
#email:String
#password:String
#phoneNum:String
1
+adminInfo(username:String,email:String,
password:String, phoneNum:String)
+setUsername(username:String):void
+setEmail(email:String):void
+setPassword(password:String):void
+ setPhoneNum(phoneNum:String):void
+getName():String
+getEmail:()String
+getPassword():String
+getUsername():String
#saveAdminDataToFile():void
<<entity>>
Address
-governorate:string
-district:string
-street:string
-buildingNumber:string
-floor:string
-flat:string
-landmark:string
1
+Address(governorate:string ,string:district,
street: string,buildingNumber: string ,floor:st
,flat:string ,landmark:string)
+setGovernorate(governorate:string):void
+setDistrict(district:string):void
+setStreet(street:string):void
+setBuildingNumber(buildingNumber:string)
+setFloor(floor:string):void
+setFlat(flat:string):void
+setLandmark(landmark:string):string
+getGovernorate():string
+getDistrict():string
+getStreet():string
+getBuildingNumber():string
+getFloor():string
+getFlat():string
+getLandmark():string
has
<<control>>
Payment
-amount:float
+Payment(amount:float)
+getAmount:float
+settlePayment():boolean
+displayMessage():void
+deductAmount():boolean
<<boundary>>
customerInfo
-address:address
-status:boolean
+customerInfo(username:String,status:boolean,address:address,
email:String,phoneNum:String,password:String)
+setAddress(address:address):void
+setStatus(status:boolean):void
+getAddress:()string
+getStatus():bool
+saveCustomerDataToFile():void
<<boundary>>
OTP
-reciverEmain:String,
+chkOTP:String
+getChkOTP():String
+OTP(email:String)
-generateOTP():String
<<entity>>
OrderStatus
OPEN:OrderStatus
CLOSED:OrderStatus
<<entity>>
paymentManager
-status:OrderStatus
-deliveryAddress:String
-payment:Payment
-orderDetails:ArrayList<Order>
-taxPercentage:float
+paymentManager(status:OrderStatus,deliveryAddress:String,payment:Payment
,orderDetails:ArrayList<Order>,taxPercentage:float,orderID:int)
+getOrderID():int
+getStatus():OrderStatus
+calculateOrderTotal():float
+payOrder(payment:Payment):boolean

CS251: Phase 2 – <Team Name>
Project: Toffee

Software Design Specification

III. Class Descriptions

Class ID	Class Name	Description & Responsibility
1.	customerInfo	The customerInfo class represents the customer's information in the system and is responsible for managing and storing customer data.The class has attributes such as password, email, phoneNum, status, username, and address, which represent various aspects of the customer's information. It also has methods such as getAddress(), setAddress(), and setStatus() that allow for getting and setting the customer's address and status. The class also has a method saveCustomerDataToFile() that saves the customer's registration data to a text file. This method is responsible for persisting customer data and allows for retrieving it later if needed.Overall, the customerInfo class is responsible for managing and storing customer information, making it an entity class.
2.	Item	The Item class represents a product in the system and is responsible for managing and storing information about the product.The class has attributes such as name, description, category, price, discount, available, and quantity.It also has methods such as getName(), getPrice(), and getQuantity() that allow for getting the product's name, price, and quantity.The class also has constructors and setter methods that allow for creating and updating the product's information. This allows for managing the product's data and keeping it up-to-date in the system.Overall, the Item class is responsible for managing and storing information about a product, making it an entity class.
3.	Catalog	The Catalog class represents the catalog of items in the system and is responsible for managing and storing information about products.The class has methods such as retrieveItemsFromFile(), updateCatalog(), and addToCatalog() that allow for retrieving, updating, and adding information about products to a text file. These methods are responsible for persisting item data and updating it in the system.The retrieveItemsFromFile() method reads the item data from a text file and creates Item objects based on the data. The updateCatalog() method updates the item data in the text file based on the changes made to the Item objects. The addToCatalog() method adds a new Item to the catalog in the text file.Overall, the Catalog class is responsible for managing and storing information about products in the system, making it an entity class.
4.	Address	This class contains all the necessary information about the user’s address which includes governorate, district, street, building information (number, floor, flat), and landmark. This class is called when the user is about to make an order and is entering his details.
5.	Order	The Order class represents an order in the system and is responsible for managing and storing information about the order.The class has methods such as getTotal(), setAddressAndEmail(), confirmOrder(), saveOrderHistory(), and getOrderHistory() that allow for calculating the total price of the order, setting the shipping address and email of the customer, confirming the order, saving order information to a text file, and retrieving order history for a specific customer.The Order class includes a shopping cart, a list of items, the total price of the order, the customer's information, and the date-time of the checkout. It also has methods to set and retrieve the order ID. Overall, the Order class is responsible for managing and storing information about orders in the system, making it an entity class.

Software Design Specification

Class ID	Class Name	Description & Responsibility
6.	Shopping Cart	The shoppingCart class represents a shopping cart in the system and is responsible for managing and storing information about the items in the cart.The class has methods such as addToCart(), removeFromCart(), displayCart(), clearCart(), saveCartToFile(), and retrieveCartItems() that allow for adding and removing items from the cart, displaying the items in the cart, clearing the cart, saving the cart information to a text file, and retrieving saved cart information for a specific customer.The shoppingCart class includes a catalog of items, a list of catalog items, and a list of cart items. It also has a total price for the items in the cart and methods to retrieve and update the quantity of items in both the catalog and cart.Overall, the shoppingCart class is responsible for managing and storing information about items in the shopping cart, making it an entity class.
7.	Payment	The Payment class is an abstract class representing a payment in the system and is responsible for managing and storing information about the payment.The class has a constructor to initialize the payment amount and methods such as getAmount(), settlePayment(), deductAmount(), and displayMessage() that allow for retrieving the payment amount, settling the payment, deducting the payment amount, and displaying a message related to the payment.The Payment class includes a payment amount and an abstract method deductAmount() which is implemented by its subclasses to deduct the payment amount from the user's account and displayMessage() which is implemented by its subclasses to display a message related to the payment.Overall, the Payment class is responsible for managing and storing information about payments in the system, making it an abstract entity class.
8.	adminInfo	The adminInfo class is responsible for storing and managing information about an administrator in the system.The class includes information such as the administrator's username, password, email, and phone number.The adminInfo class has methods such as saveAdminDataToFile() which saves the administrator's information to a text file named adminInfo.txt. This method writes the administrator's username, password, email, and phone number to the text file.Overall, the adminInfo class is responsible for managing and storing information about the administrator in the system, making it an entity class.
9.	OTP	The OTP class is responsible for generating and sending a One-Time Password (OTP) to a user's email address for verification purposes.The class includes information such as the recipient email address and the generated OTP. The OTP class has methods such as sendOTP() which generates an OTP and sends it to the recipient email address using the JavaMail API. This method uses the email and password of the sender's email account to authenticate and send the email.The generateOTP() method is used to generate a random 6-digit OTP for verification purposes.Overall, the OTP class is responsible for generating and sending OTPs for user verification, making it a utility class.
10.	OrderStatus	The OrderStatus enum is responsible for defining the possible status values for an order in the system.The enum includes two possible values: OPEN and CLOSED.The OPEN status indicates that the order is currently open and has not been paid for yet, while the CLOSED status indicates that the order has been paid for and is now closed. Overall, the OrderStatus enum is responsible for defining the possible status values for an order in the system, making it a utility class.

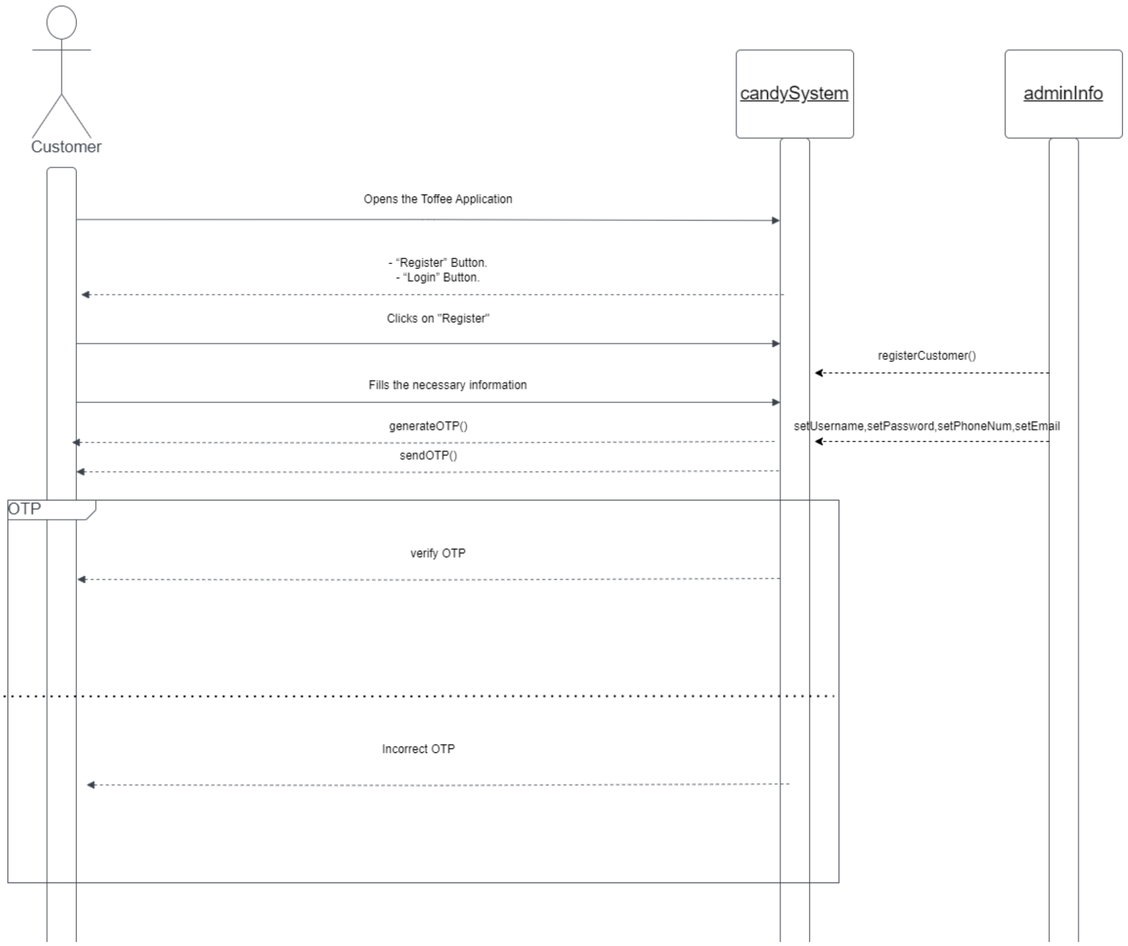
Software Design Specification

Class ID	Class Name	Description & Responsibility
11.	paymentManager	The paymentManager class is responsible for managing and processing payments for orders in the system.The class includes information such as order status, order total, delivery address, payment, order details, tax percentage, and order ID.The paymentManager class has methods such as calculateOrderTotal(), which calculates the total cost of the order by iterating through the list of order details and adding the total cost of each order item with tax percentage applied.The payOrder() method is used to settle the payment for the current order. It first checks if the order has already been paid for, and if not, it calls the settlePayment() method of the Payment object to process the payment. If the payment is successful, the order status is set to closed, and the method returns true.Overall, the paymentManager class is responsible for managing and processing payments for orders in the system, making it an entity class.
12.	Main	The Main class is responsible for controlling the flow of the candy system application. The class includes a main method that displays a menu of options for the user to choose from. The options include registering as a customer or an admin, logging in, and performing various actions depending on the user type. The Main class uses an instance of the candySystem class to perform actions such as registering a new user, logging in, adding items to the catalog, adding items to the shopping cart, making an order, and paying for the order.Overall, the Main class is responsible for controlling the flow of the candy system application and delegating tasks to other classes, making it a control class.

Software Design Specification

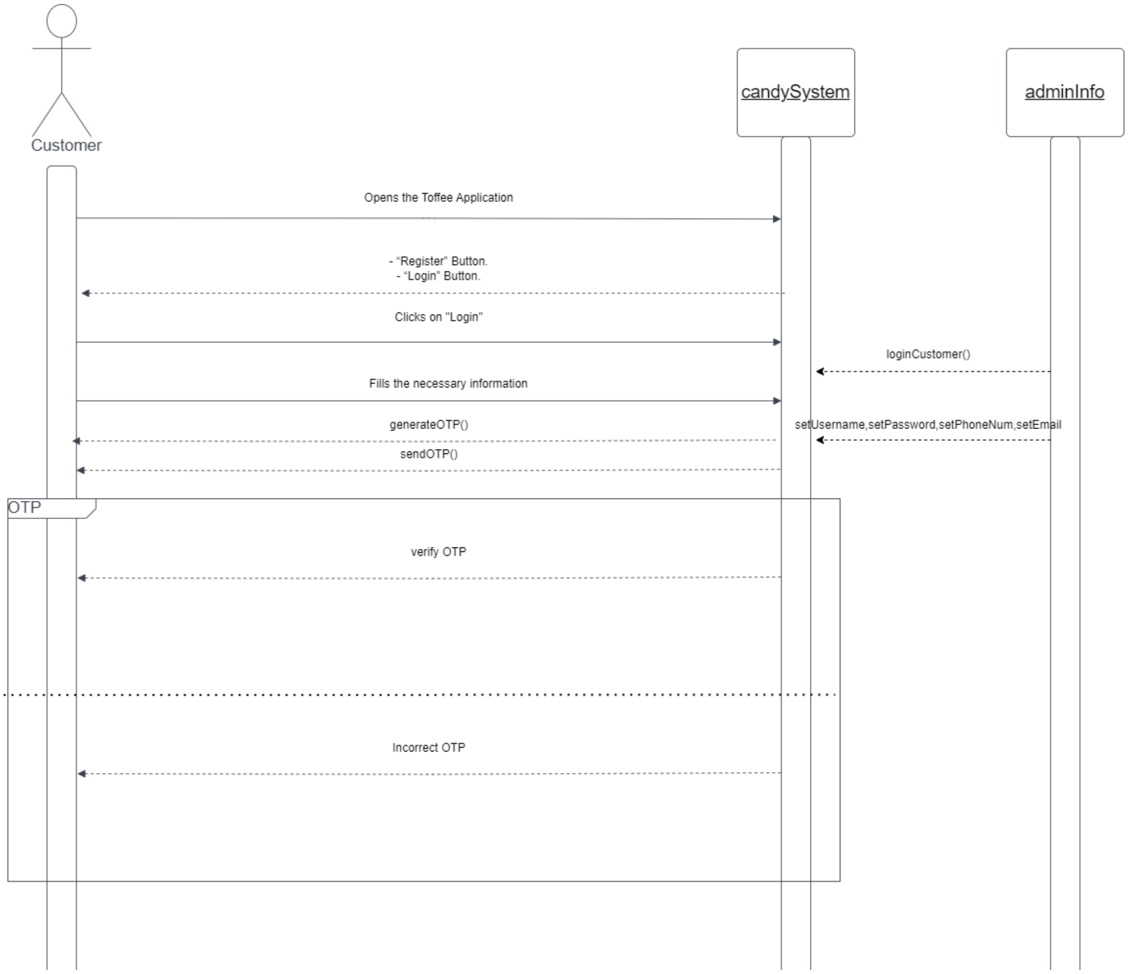
IV. Sequence diagrams

1. Register’s sequence diagram:



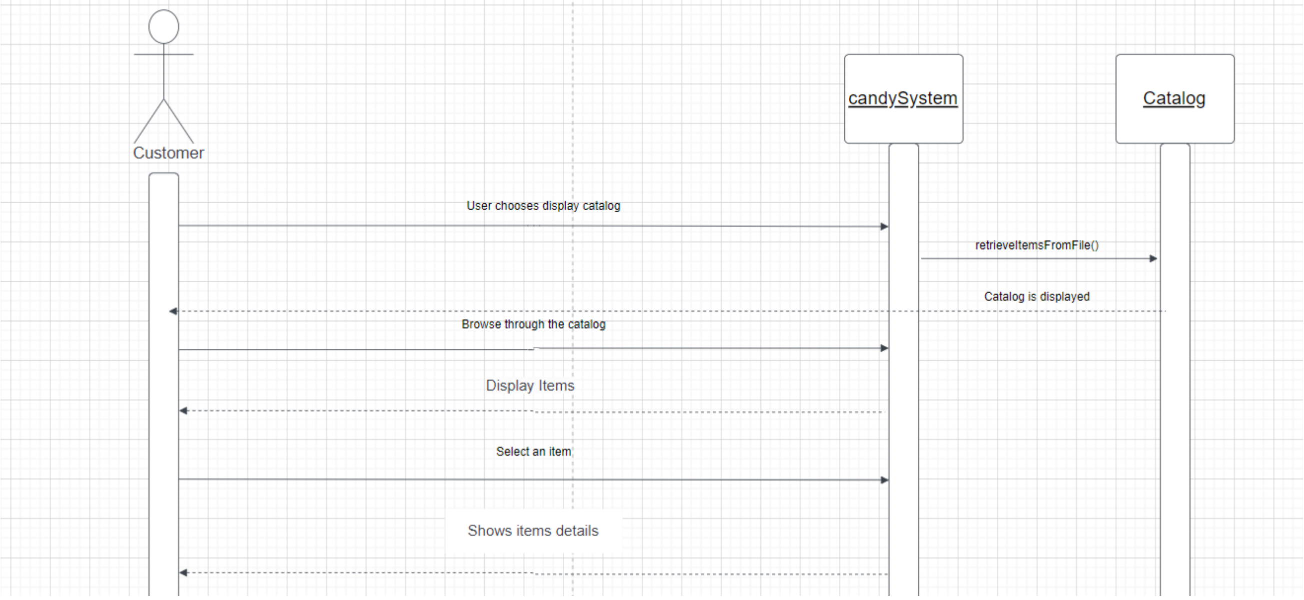
Software Design Specification

2. Login’s sequence diagram

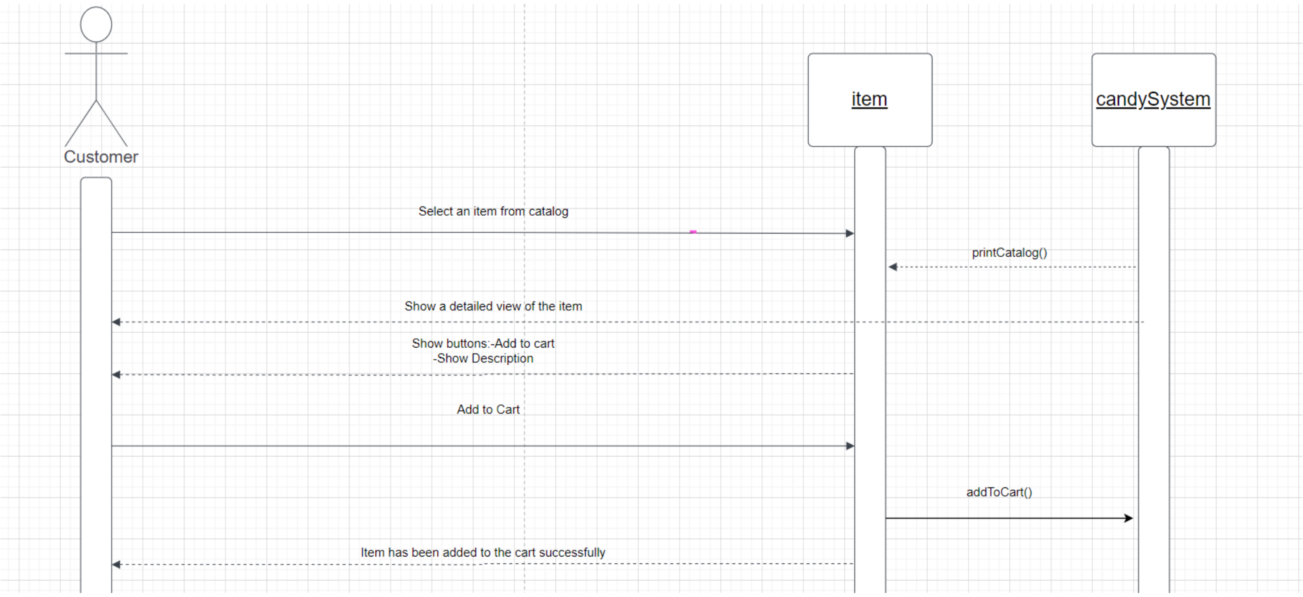


Software Design Specification

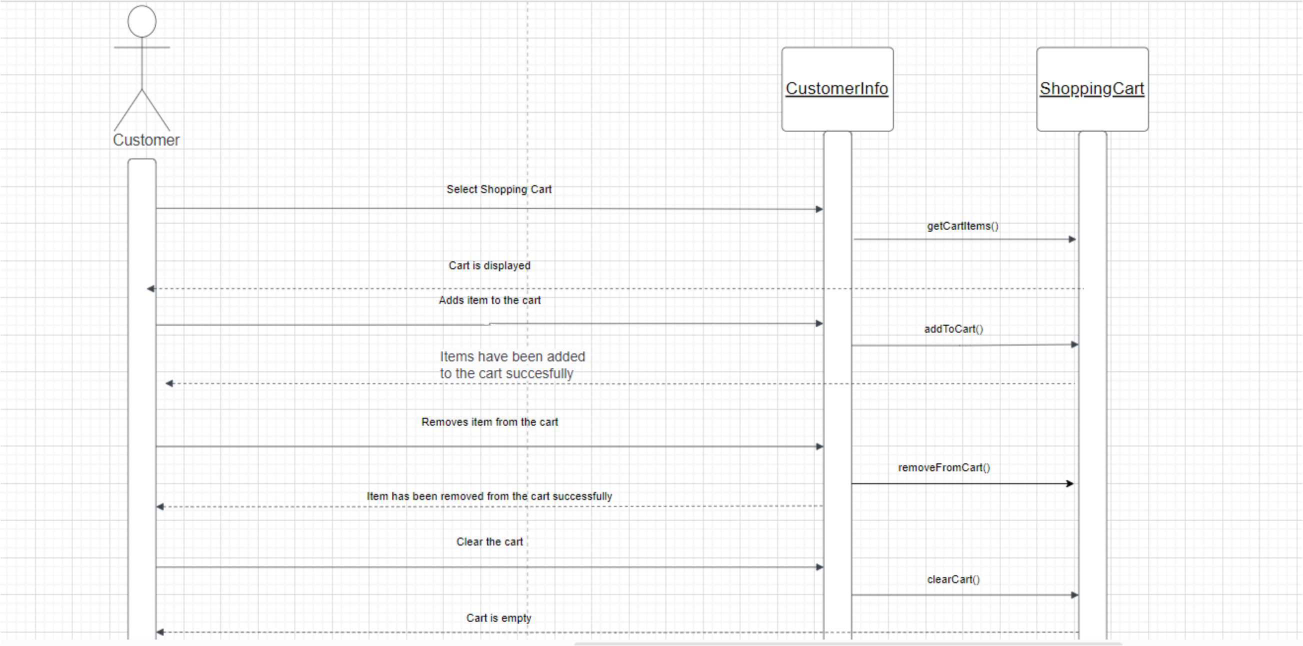
3. Browse Items Catalog’s Sequence Diagram



4. Select an item from Catalog’s Sequence Diagram:

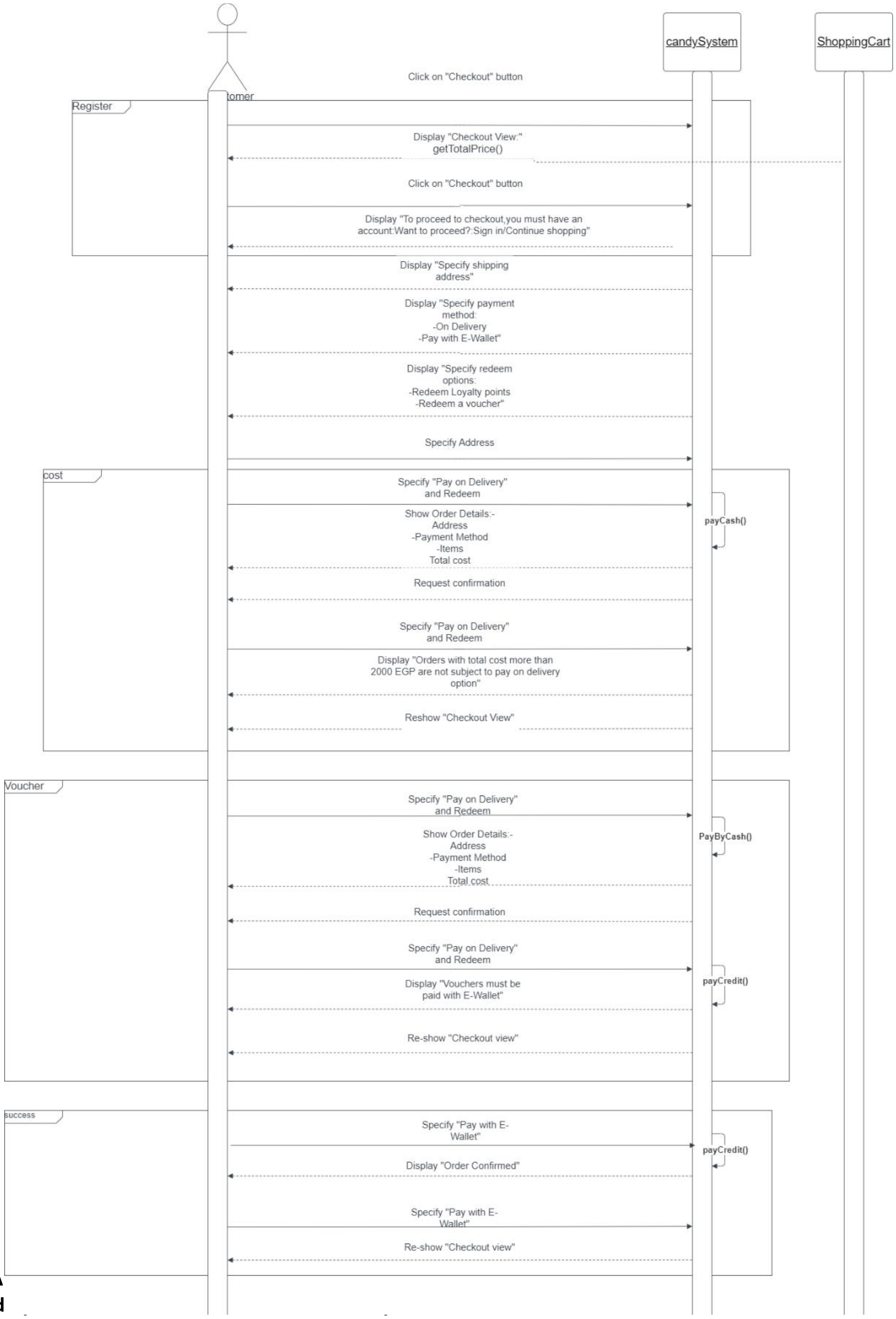

Customer
item
candySystem
Select an item from catalog
printCatalog()
Show a detailed view of the item
Show buttons:-Add to cart
-Show Description
Add to Cart
addToCart()
Item has been added to the cart successfully

5. View Shopping Cart's Sequence Diagram



6. Checkout's Sequence Diagram

Software Design Specification


candySystem
ShoppingCart
tomer
Click on "Checkout" button
Register
Display "Checkout View:"
getTotalPrice()
Click on "Checkout" button
Display "To proceed to checkout,you must have an account:Want to proceed?:Sign in/Continue shopping"
Display "Specify shipping address"
Display "Specify payment method:
-On Delivery
-Pay with E-Wallet"
Display "Specify redeem options:
-Redeem Loyalty points
-Redeem a voucher"
Specify Address
cost
Specify "Pay on Delivery" and Redeem
payCash()
Show Order Details:-
Address
-Payment Method
-Items
Total cost
Request confirmation
Specify "Pay on Delivery" and Redeem
Display "Orders with total cost more than 2000 EGP are not subject to pay on delivery option"
Reshow "Checkout View"
Voucher
Specify "Pay on Delivery" and Redeem
PayByCash()
Show Order Details:-
Address
-Payment Method
-Items
Total cost
Request confirmation
Specify "Pay on Delivery" and Redeem
payCredit()
Display "Vouchers must be paid with E-Wallet"
Re-show "Checkout view"
success
Specify "Pay with E-Wallet"
payCredit()
Display "Order Confirmed"
Specify "Pay with E-Wallet"
Re-show "Checkout view"

Software Design Specification

Class - Sequence Usage Table

Sequence Diagram	Classes Used	All Methods Used
1. Register	User OTP adminInfo	OTP:generateOTP() OTP:sendOTP adminInfo:registerCustomer,setters
2. Login	candySystem OTP adminInfo	candySystem:loginCustomer,setters OTP:generateOTP() OTP:sendOTP
3. Browse Items Catalog	candySystem	retrieveItemsFromFile()

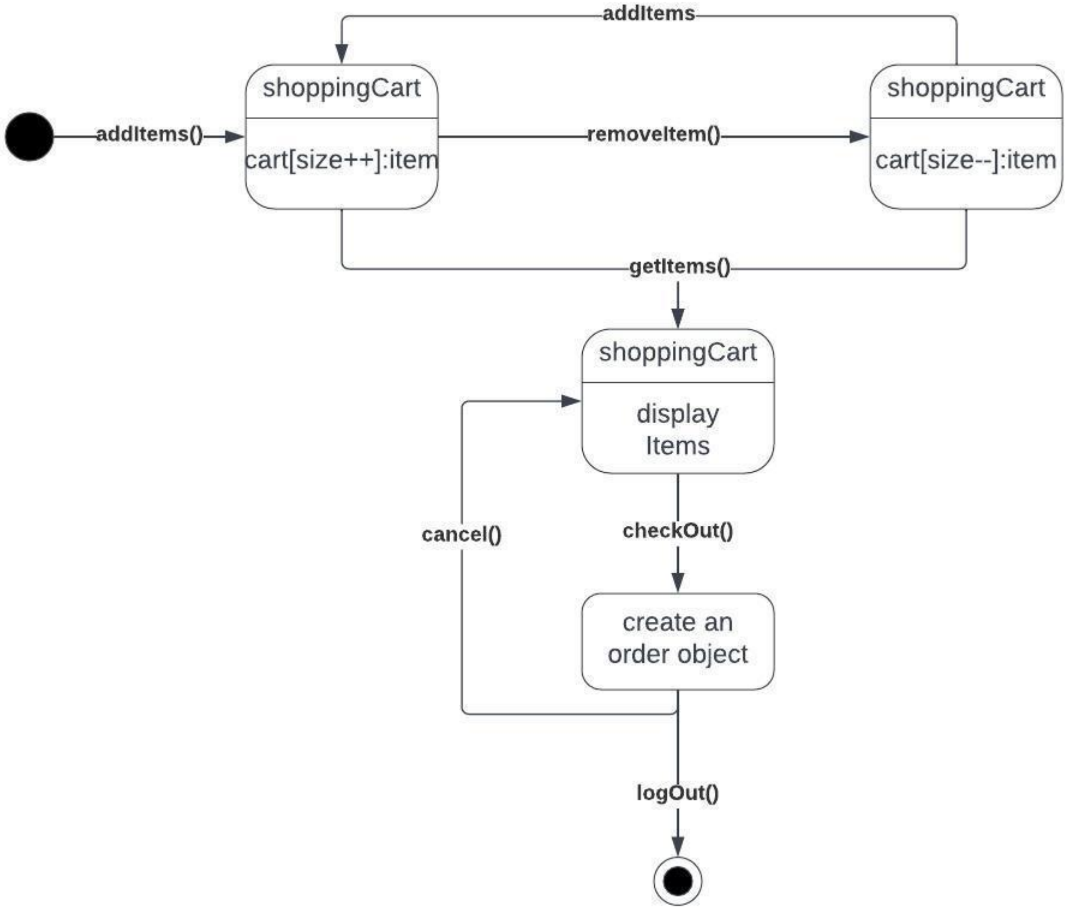
4. Select item from catalog	candySystem	printCatalog()
		addToCart()

5. View Shopping Cart	ShoppingCart	removeFromCart()

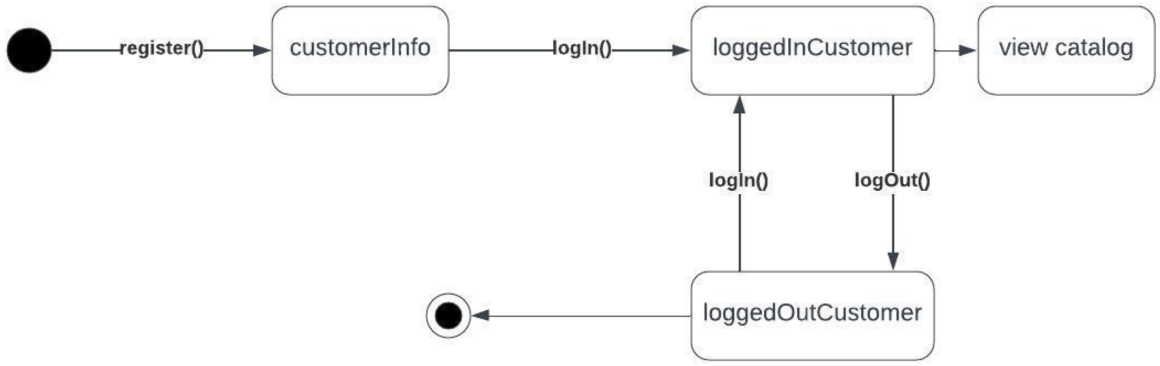
6. Checkout	Payment	payCash()
		payCredit()

Software Design Specification

V. State Diagram



Software Design Specification



Tools

- LucidChart
- Draw.io

Ownership Report

Item	Owners
Laila Hesham Kandil	Part of the class diagram, part of the class description, and 6 sequence diagrams and sequence usage table and part of the code
Youssef Ahmed Zakaria	Document's Purpose and Audience and the architecture diagram and part of the code
Doaa Ali ElSayed	Part of the class diagram and part of the class description and state diagram and part of the code

The Github Repo: https://github.com/LailaKandil/SW-A3