Cairo University **Faculty of Computers and Artificial Intelligence**

Software design specification document

2022

Project Team

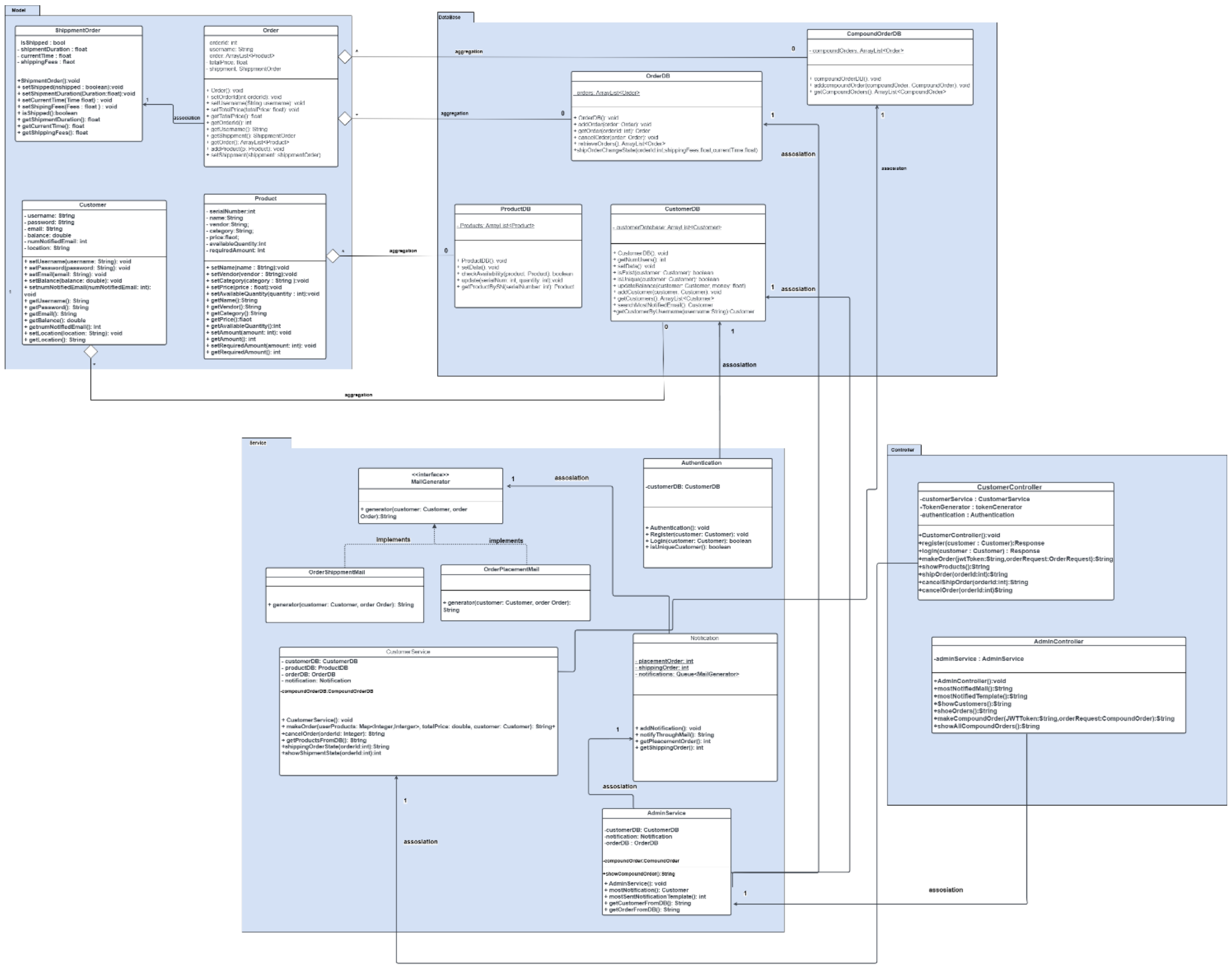
ID	Name	Email
20210063	Esraa Mahmoud Abd El Mohsen	Esraamahmoud2334@gmail.com
20210235	Abdallah Hussein Ibrahim Hussein	abdallah.hus74@gmail.com
20210205	Abdelrhman Sayed Ali	
20210019	Ahmad Reda Bayoumy	

Contents

CS352: Sprint SDS– Advangers-

SDS document

Class diagram design



Model
ShippmentOrder
IsShipped : bool
- shipmentDuration : float
- currentTime : float
- shippingFees : float
+ShipmentOrder():void
+ setShipped(nshipped : boolean):void
+ setShipmentDuration(Duration:float):void
+ setCurrentTime(Time float) : void
+ setShipingFees(Fees : float) : void
+ isShipped():boolean
+ getShipmentDuration(): float
+ getCurrentTime(): float
+ getShippingFees(): float
Order
orderId: int
username: String
order: ArrayList<Product>
- totalPrice: float
- shipment: ShippmentOrder
+ Order(): void
+ setOrderId(int orderId): void
+ setUsername(String username): void
+ setTotalPrice(totalPrice: float): void
+ getTotalPrice(): float
+ getOrderId(): int
+ getUsername(): String
+ getShippment(): ShippmentOrder
+ getOrder(): ArrayList<Product>
+ addProduct(p: Product): void
+ setShippment(shippment: shippmentOrder)
association
1
Customer
- username: String
- password: String
- email: String
- balance: double
- numNotifiedEmail: int
- location: String
+ setUsername(username: String): void
+ setPassword(password: String): void
+ setEmail(email: String): void
+ setBalance(balance: double): void
+ setnumNotifiedEmail(numNotifiedEmail: int): void
+ getUsername(): String
+ getPassword(): String
+ getEmail(): String
+ getBalance(): double
+ getnumNotifiedEmail(): int
+ setLocation(location: String): void
+ getLocation(): String
Product
- serialNumber:int
- name:String
- vendor:String;
- category:String;
- price:flaot;
- availableQuantity:int
- requiredAmount: int
+ setName(name : String):void
+ setVendor(vendor : String):void
+ setCategory(category : String):void
+ setPrice(price : float):void
+ setAvailableQuantity(quantity : int):void
+ getName():String
+ getVendor():String
+ getCategory():String
+ getPrice():flaot
+ getAvailableQuantity():int
+ setAmount(amount: int): void
+ getAmount(): int
+ setRequiredAmount(amount: int): void
+ getRequiredAmount(): int
aggregation
*
0
DataBase
CompoundOrderDB
- compoundOrders: ArrayList<Order>
+ compoundOrderDB(): void
+ addcompoundOrder(compoundOrder: CompoundOrder): void
+ getCompoundOrders(): ArrayList<CompoundOrder>
OrderDB
- orders: ArrayList<Order>
+ OrderDB(): void
+ addOrder(order: Order): void
+ getOrder(orderId: int): Order
+ cancelOrder(order: Order): void
+ retrieveOrders(): ArrayList<Order>
+shipOrderChangeState(orderId:int,shippingFees:float,currentTime:float)
assosiation
ProductDB
- Products: ArrayList<Product>
+ ProductDB(): void
+ setData(): void
+ checkAvailability(product: Product): boolean
+ update(serialNum: int, quantity: int):void
+ getProductBySN(serialNumber: int): Product
CustomerDB
- customerDatabase: ArrayList<Customer>
+ CustomerDB(): void
+ getNumUsers(): int
+ setData(): void
+ isExist(customer: Customer): boolean
+ isUnique(customer: Customer): boolean
+ updateBalance(customer: Customer, money: float)
+ addCustomer(customer: Customer): void
+ getCustomers(): ArrayList<Customer>
+ searchMostNotifiedEmail(): Customer
+getCustomerByUsername(username:String):Customer
Service
<<interface>>
MailGenerator
+ generator(customer: Customer, order Order):String
implements
OrderShippmentMail
+ generator(customer: Customer, order Order): String
OrderPlacementMail
+ generator(customer: Customer, order Order): String
Authentication
-customerDB: CustomerDB
+ Authentication(): void
+ Register(customer: Customer): void
+ Login(customer: Customer): boolean
+ isUniqueCustomer(): boolean
CustomerService
- customerDB: CustomerDB
- productDB: ProductDB
- orderDB: OrderDB
- notification: Notification
-compoundOrderDB:CompoundOrderDB
+ CustomerService(): void
+ makeOrder(userProducts: Map<Integer,Interger>, totalPrice: double, customer: Customer): String+
+cancelOrder(orderId: Integer): String
+ getProductsFromDB(): String
+shippingOrderState(orderId:int):String
+showShipmentState(orderId:int):int
Notification
- placementOrder: int
- shippingOrder: int
- notifications: Queue<MailGenerator>
+ addNotification(): void
+ notifyThroughMail(): String
+ getPleacementOrder(): int
+ getShippingOrder(): int
AdminService
-customerDB: CustomerDB
-notification: Notification
-orderDB : OrderDB
-compoundOrder:ComoundOrder
+showCompoundOrder():String
+ AdminService(): void
+ mostNotification(): Customer
+ mostSentNotificationTemplate(): int
+ getCustomerFromDB(): String
+ getOrderFromDB(): String
Controller
CustomerController
-customerService : CustomerService
-TokenGenerator : tokenGenerator
-authentication : Authentication
+CustomerController():void
+register(customer : Customer):Response
+login(customer : Customer) : Response
+makeOrder(jwtToken:String,orderRequest:OrderRequest):String
+showProducts():String
+shipOrder(orderId:int):String
+cancelShipOrder(orderId:int):String
+cancelOrder(orderId:int)String
AdminController
-adminService : AdminService
+AdminController():void
+mostNotifiedMail():String
+mostNotifiedTemplate():String
+ShowCustomers():String
+shoeOrders():String
+makeCompoundOrder(JWTToken:String,orderRequest:CompoundOrder):String
+showAllCompoundOrders():String
assosiation

SDS document

Class diagram Explanation

- Façade Design Pattern
- **This pattern is used for separating the complex subsystem and make it easier to use by implementing a Facade class (classes in controller package) that provides one, more reasonable interface.**
- **The classes participate (classes in controller package, classes in services package, classes in Model package)**
- Composite design pattern
- **This pattern used because can make order or composite order that consist of multiple order**
- **The classes participate (compoundOrderDB, Order)**
- Strategy design pattern
- **This pattern used because the same thing (generate mail) can done by different way**
- **The classes participate (MailGenerator,OrderPlacementMail,OrderShippmentMail)**

Requirements Exposure as Web Service API

Part 1: Exposed Postman Collection

https://orange-water-528672.postman.co/workspace/My-Workspace~80e368d2-49f0-4b98-81d2-7d781c57d032/collection/31628756-c6752499-18f7-4398-8129-665769a0c56c?action=share&creator=31628756

Part 2:

Requirement		**Exposed API**
The system should list of all the products currently available for purchase should be displayed.		1- CustomerControllerAPI 2- GET /ShowProducts 3- showProducts()→operation
create an account and put a specific balance using that account.		1- CustomerControllerAPI 2- POST/login 3- login()àOperation

CS352: Sprint SDS– Advangers-

SDS document

Customer can login to the system		1- CustomerControllerAPI 2- POST/register 3- Register()->Operation
customer can place a simple order		1- CustomerControllerAPI 2- POST/makeOrder 3- makeOrder()→operation
customer can place a compound order		1-CustomerControllerAPI 2-POST/makeCompoundOrder 3- makeCompoundOrder()→Operation
list all the details of compound orders		1- AdminControllerAPI 2- GET/ShowCompoundOrder 3- showAllCompoundOrder()- >operation
list all the details of simple orders		1- AdminControllerAPI 2- GET/ ShowOrders 3- showOrders())-> operation
Ship a compound order		1- CustomerControllerAPI 2- POST\ShippingOrder 3- shipOrder()→operation
Ship the order		1- CustomerControllerAPI 2- POST\ShippingOrder 3- shipOrder()→operation
Notifications creation for various operations.		1- CustomerControllerAPI 2- Created in makeOrder()function when user make order 3- POST/makeOrder
cancel only its shipping		1- CustomerControllerAPI 2- POST/CancelShippingOrder 3- cancelShipOrder()→operation
Customers can cancel an order placement		1- CustomerControllerAPI 2- POST/CancelOrder

SDS document

		3- cancelOrder()→operation
Notifications Queue		Exist in class notification that exist in package service This notification in 2 cases • when make order ; make notification of type(OrderPlacementMail) • when ship the order; make notification of type(OrderShipmentMail)
The most notified email-address/phone-number.		1-AdminControllerAPI 2-GET/mostNotifiedCustomer mostNotifiedMail()àoperation
The most sent notification template.		1-AdminControllerAPI 2-GET /mostNotifiedTemplate mostNotifiedTempalte()→operation
Admin can show all customers		1-AdminControllerAPI 2-GET /ShowCustomers showCustomers()àoperation
Admin can show all orders		1-AdminControllerAPI 2-GET /ShowOrders showOrders()àoperation
Admin can show order shippmemtState		1-AdminControllerAPI 2-POST /ShowOrderShippmentState showOrderShipmentState()àoperation