FUTURE
UNIVERSITY IN EGYPT
جـامعـة المسـتقبـل

DAY AND NIGHT CAFÉ DNC

SOFTWARE ENGINEERING 2

Prepared by

Mahmoud Mohamed Mahmoud	20182987
Beshoy Hany Helmy	20184541
Ahmed Magdy Tawfiq	20185011
Aya Amr Abdelfattah	20190218
Habiba Gamal Abdelsalam	20193966

Supervised by

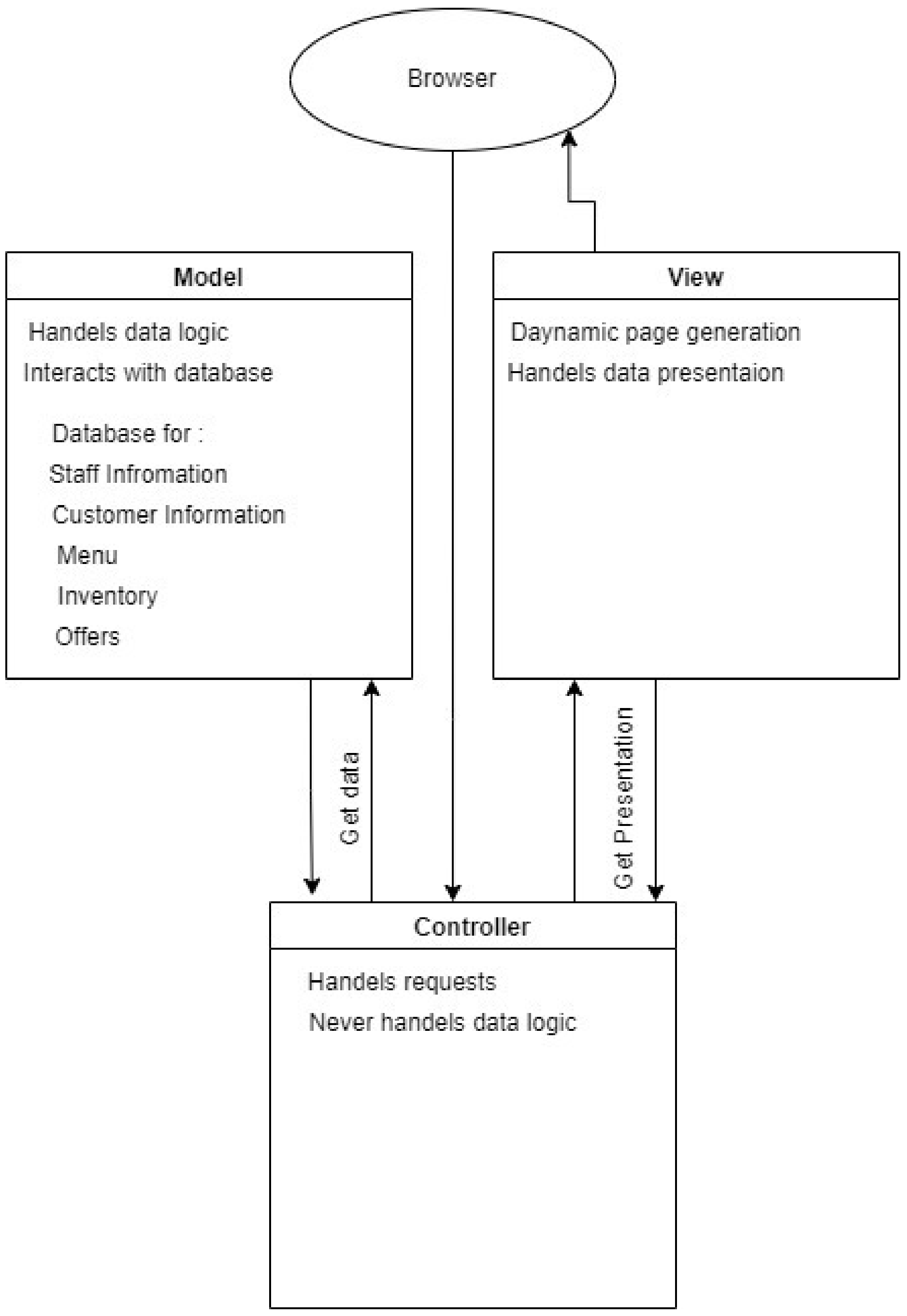
Doctor Ramadan Moawad

TA. Amr Mansour

Contents

Software Architecture

Model view controller (MVC)



MVC is an architectural pattern consisting of 3 parts:

MODEL	VIEW	Controller
Handles data logic	It displays the information from the model to the user	It controls the data flow into a model object and updates the view whenever data changes

Features of MVC:

- Easy
- Highly testable
- Extensible
- pluggable framework
- Supports for Test Driven Development (TDD)
- To design a web application architecture using the MVC pattern, it offers full control over your HTML as well as your URLs

Advantages of MVC:

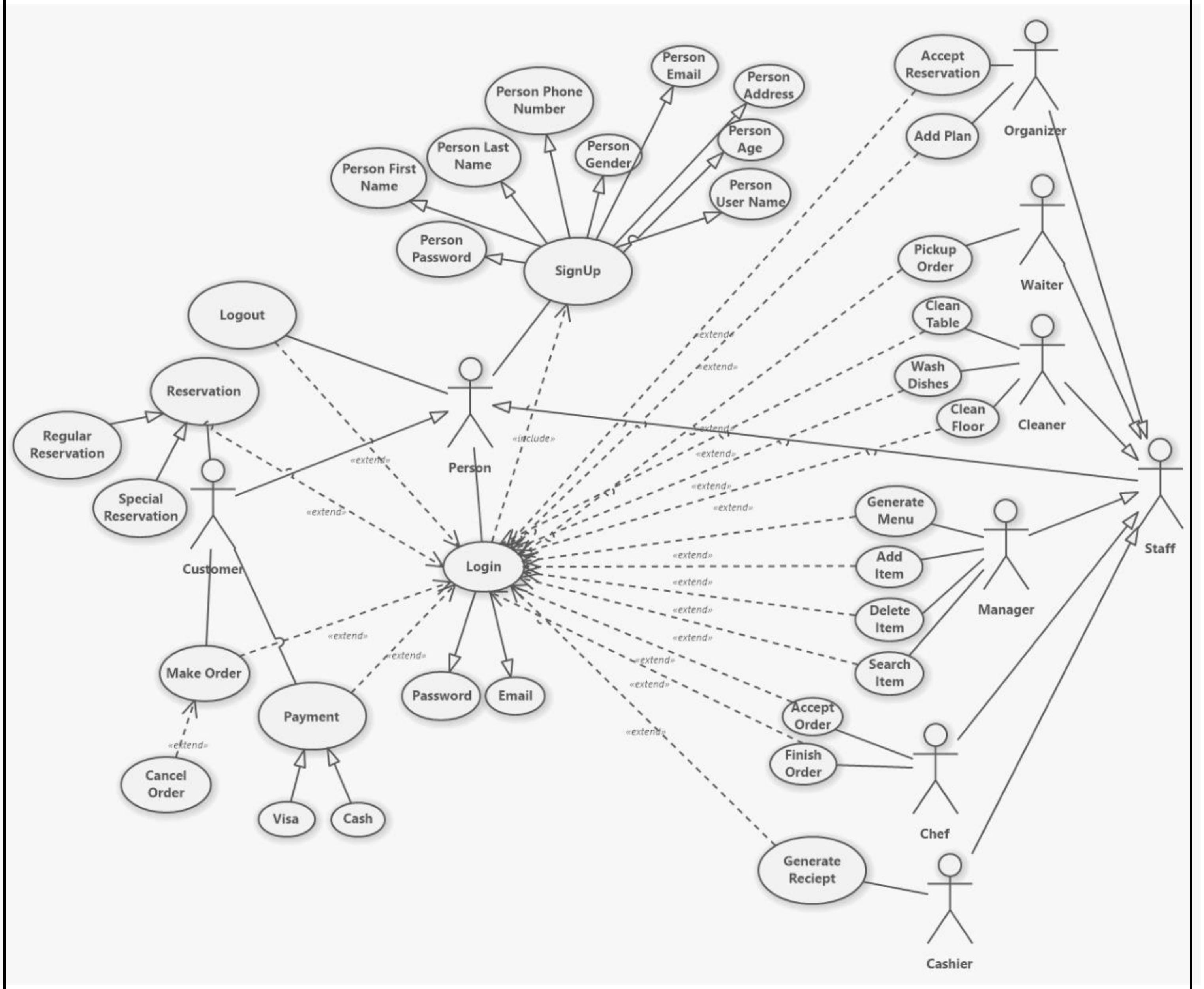
- Easy code maintenance which is easy to extend and grow
- MVC Model component can be tested separately from the user
- Easier support for new types of clients
- Development of the various components can be performed parallelly.
- It helps you to avoid complexity by dividing an application into the three units. Model, view, and controller
- It only uses a Front Controller pattern which process web application requests through a single controller.
- Offers the best support for test-driven development
- It works well for Web apps which are supported by large teams of web designers and developers.
- All classes and objects are independent of each other so that you can test them separately.

  MVC design pattern allows logical grouping of related actions on a controller together

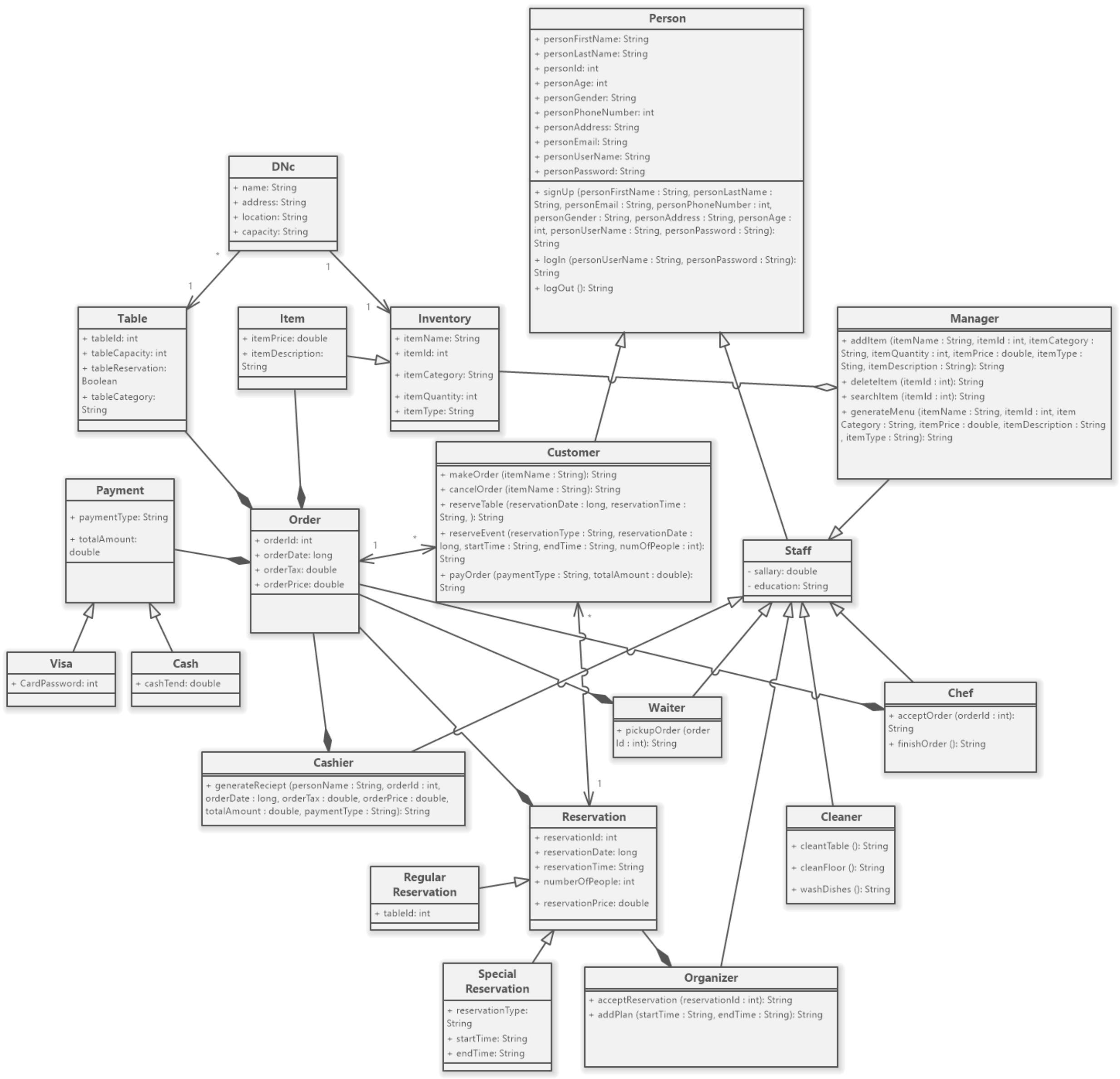
Why MVC??

- Deal with our data separately as model level doesn’t see the view level or controller level
- If any change occurred in model level, it automatically notifies observers with the changes
- Customers only deal with the view level as it’s the interface which represents the desired data by order of controller and model.
- The view also handles the requests from customers or users and informs it to the controller.
- The view also communicates with model to represent desired data to customers or users
- The controller level acts as the brain as it links the customer or the user with the system
- Controller level provides the customer and users with appropriate views and data on the screen.

USE CASE DIAGRAM

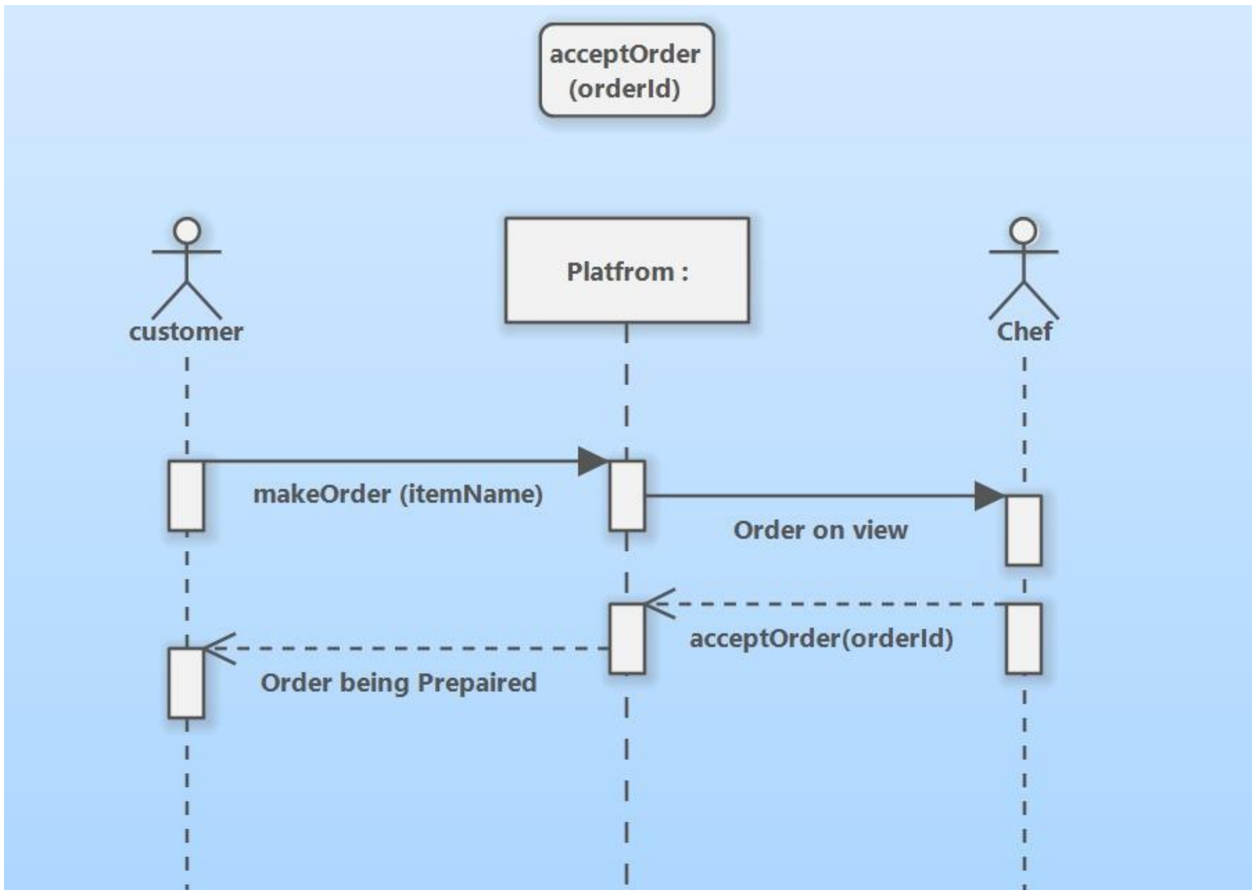


CLASS DIAGRAM

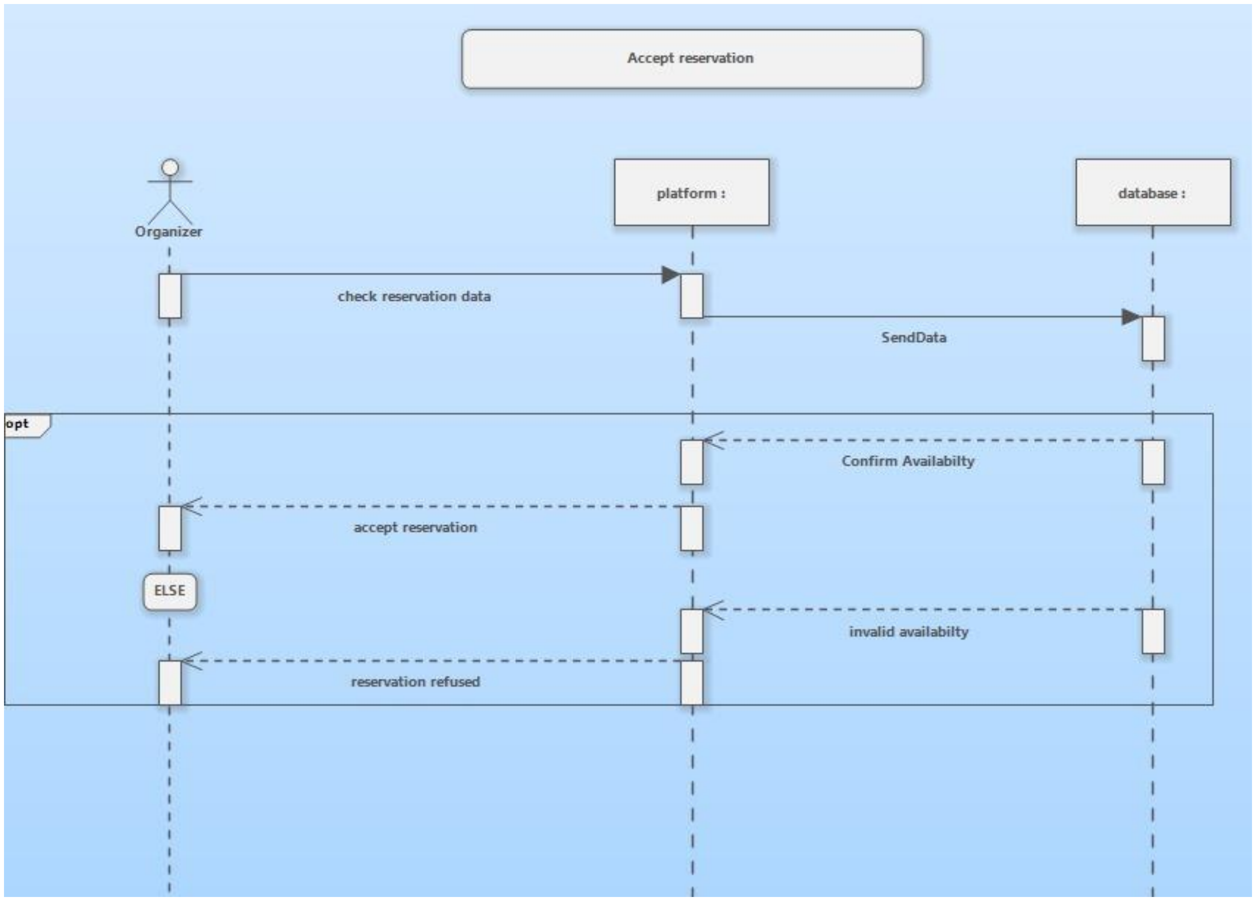

Person
+ personFirstName: String
+ personLastName: String
+ personId: int
+ personAge: int
+ personGender: String
+ personPhoneNumber: int
+ personAddress: String
+ personEmail: String
+ personUserName: String
+ personPassword: String
+ signUp (personFirstName : String, personLastName : String, personEmail : String, personPhoneNumber : int, personGender : String, personAddress : String, personAge : int, personUserName : String, personPassword : String): String
+ logIn (personUserName : String, personPassword : String): String
+ logOut (): String
DNc
+ name: String
+ address: String
+ location: String
+ capacity: String
Table
+ tableId: int
+ tableCapacity: int
+ tableReservation: Boolean
+ tableCategory: String
Item
+ itemPrice: double
+ itemDescription: String
Inventory
+ itemName: String
+ itemId: int
+ itemCategory: String
+ itemQuantity: int
+ itemType: String
Manager
+ addItem (itemName : String, itemId : int, itemCategory : String, itemQuantity : int, itemPrice : double, itemType : String, itemDescription : String): String
+ deleteItem (itemId : int): String
+ searchItem (itemId : int): String
+ generateMenu (itemName : String, itemId : int, item Category : String, itemPrice : double, itemDescription : String , itemType : String): String
Customer
+ makeOrder (itemName : String): String
+ cancelOrder (itemName : String): String
+ reserveTable (reservationDate : long, reservationTime : String,): String
+ reserveEvent (reservationType : String, reservationDate : long, startTime : String, endTime : String, numOfPeople : int): String
+ payOrder (paymentType : String, totalAmount : double): String
Payment
+ paymentType: String
+ totalAmount: double
Order
+ orderId: int
+ orderDate: long
+ orderTax: double
+ orderPrice: double
Staff
- sallary: double
- education: String
Visa
+ CardPassword: int
Cash
+ cashTend: double
Waiter
+ pickupOrder (order Id : int): String
Chef
+ acceptOrder (orderId : int): String
+ finishOrder (): String
Cashier
+ generateReciept (personName : String, orderId : int, orderDate : long, orderTax : double, orderPrice : double, totalAmount : double, paymentType : String): String
Reservation
+ reservationId: int
+ reservationDate: long
+ reservationTime: String
+ numberOfPeople: int
+ reservationPrice: double
Cleaner
+ cleanTable (): String
+ cleanFloor (): String
+ washDishes (): String
Regular Reservation
+ tableId: int
Special Reservation
+ reservationType: String
+ startTime: String
+ endTime: String
Organizer
+ acceptReservation (reservationId : int): String
+ addPlan (startTime : String, endTime : String): String
*
1
1
1
1
*
*
1

Sequence diagram

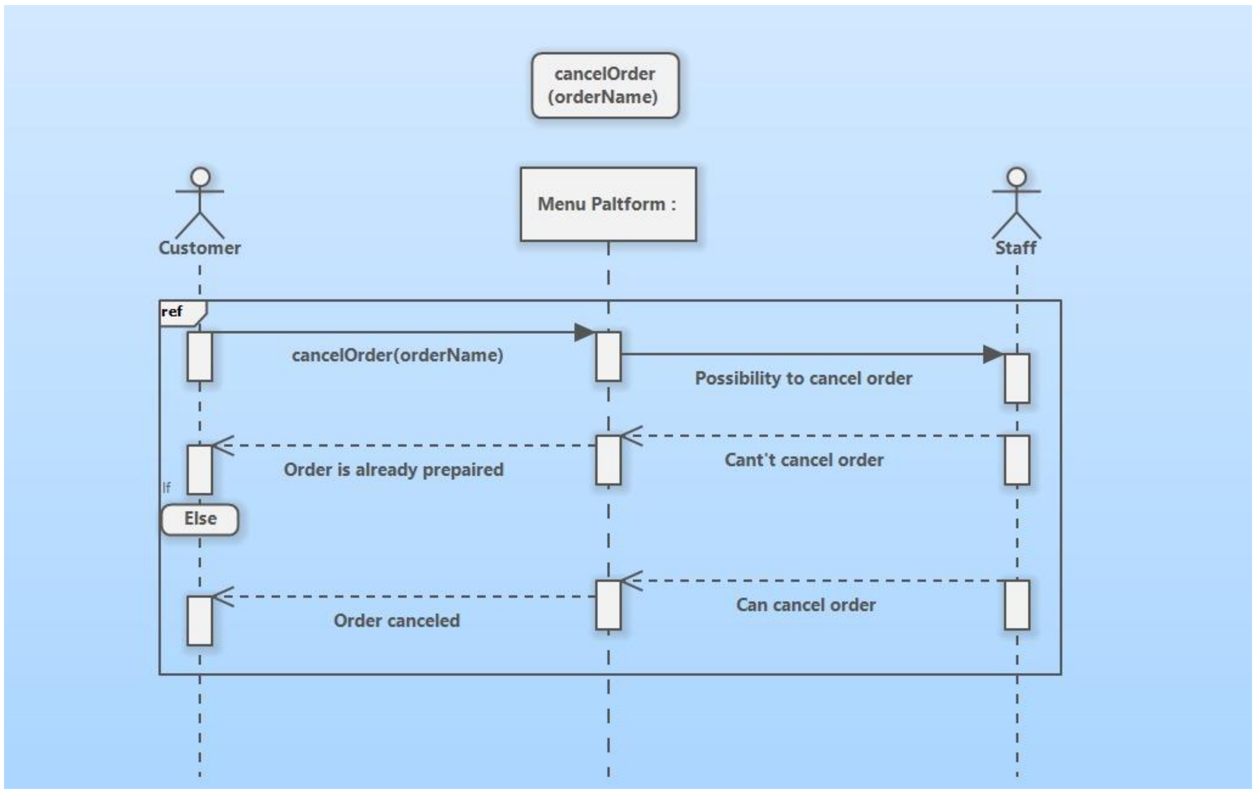
Accept Order



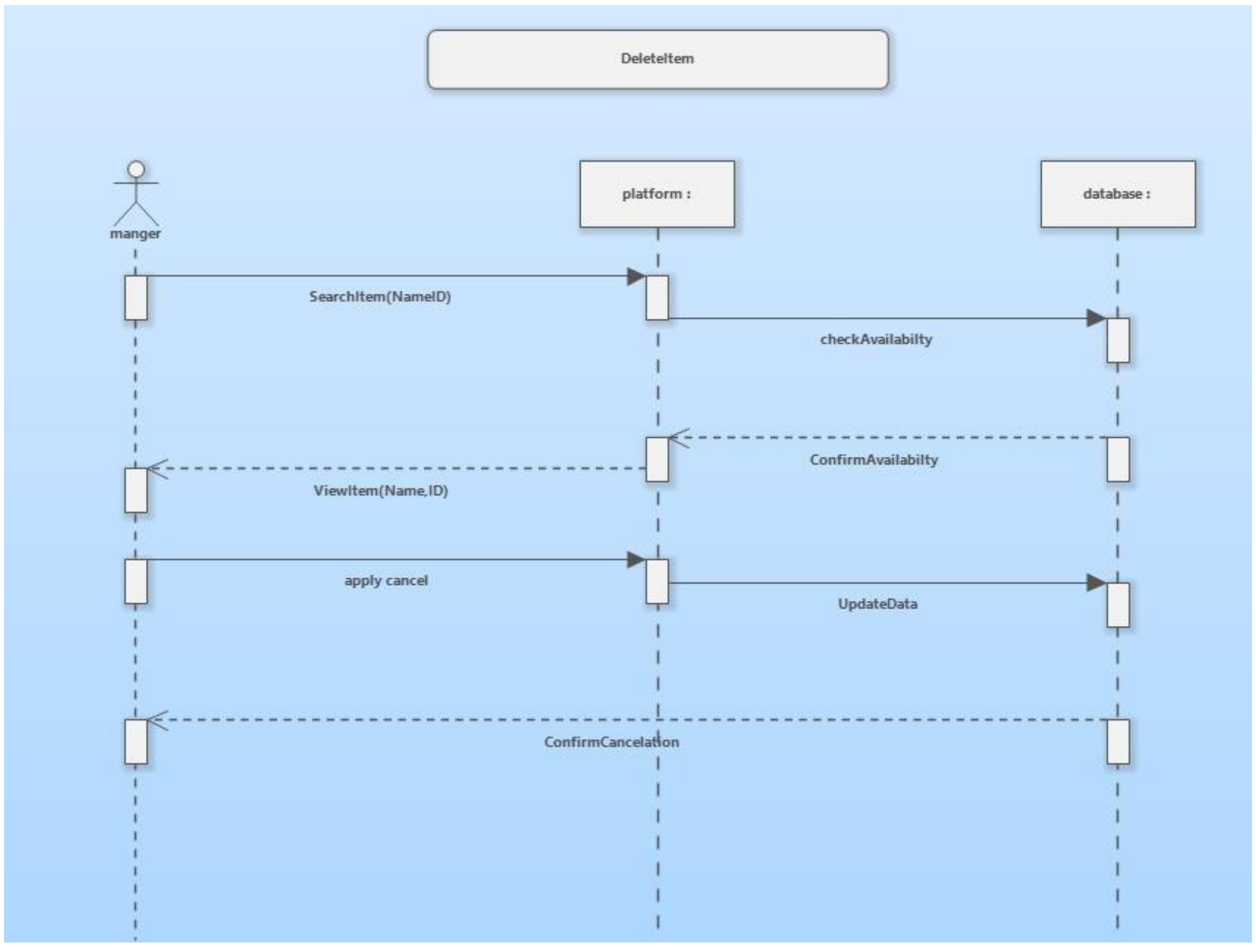
Accept reservation



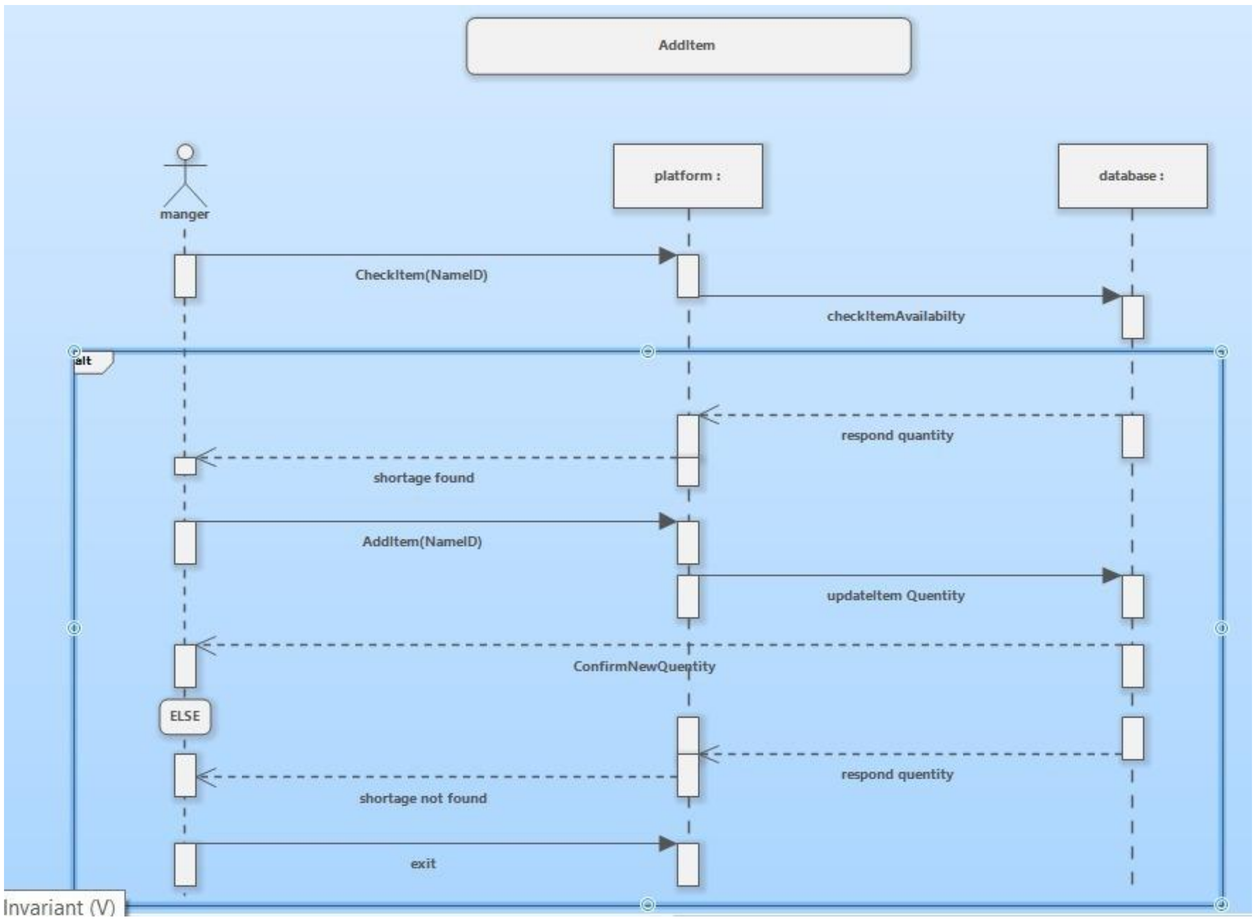
Cancel Order



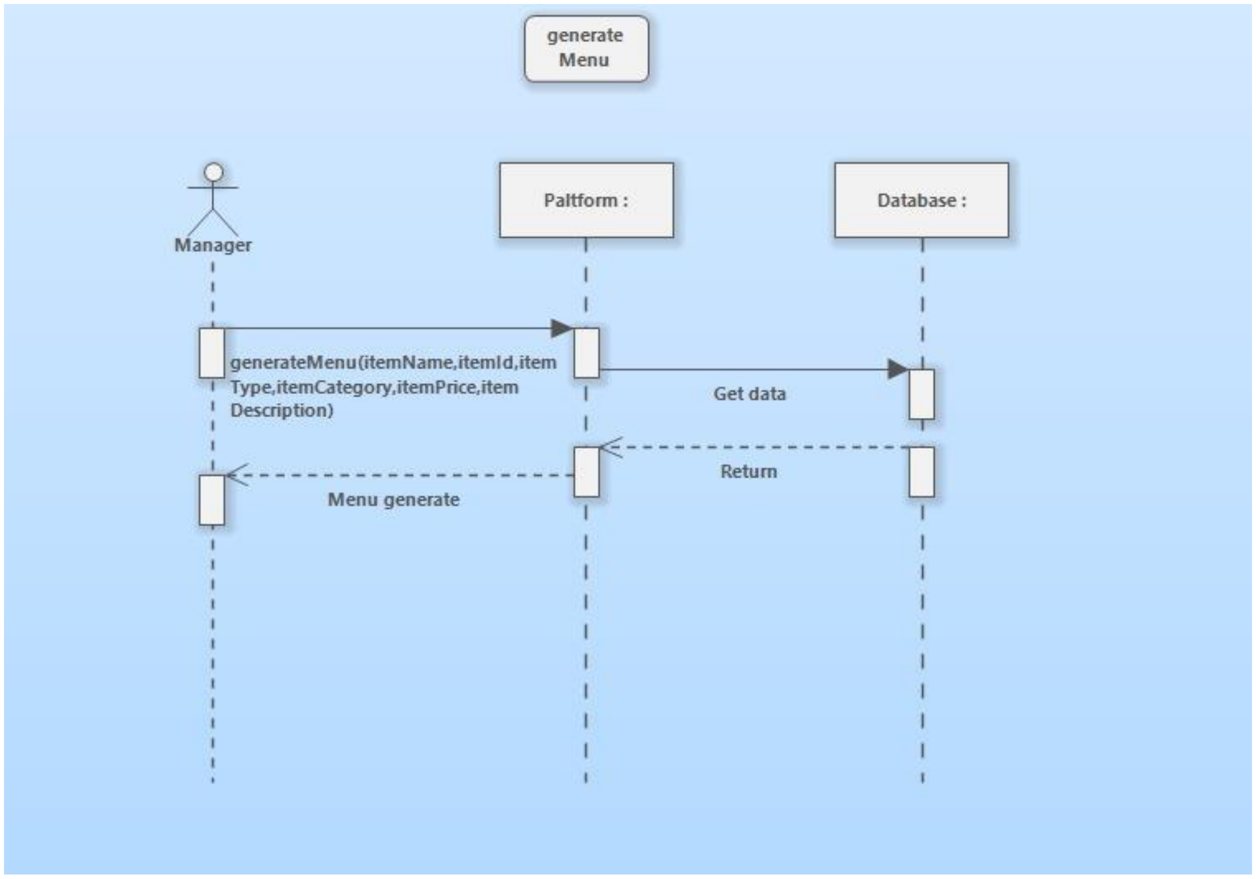
Delete Item


DeleteItem
manger
platform :
database :
SearchItem(NameID)
checkAvailabilty
ConfirmAvailabilty
ViewItem(Name,ID)
apply cancel
UpdateData
ConfirmCancelation

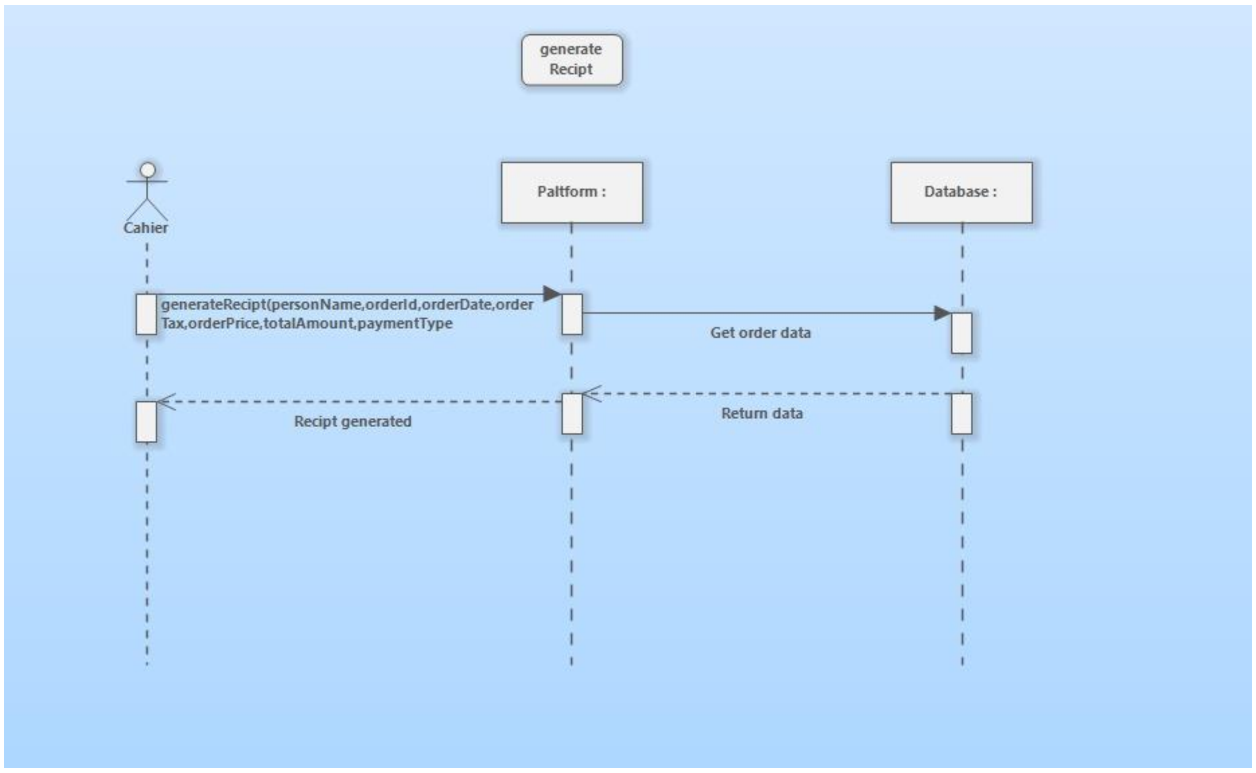
Add Item


AddItem
manger
platform :
database :
CheckItem(NameID)
checkItemAvailabilty
alt
respond quantity
shortage found
AddItem(NameID)
updateItem Quentity
ConfirmNewQuentity
ELSE
respond quentity
shortage not found
exit
Invariant (V)

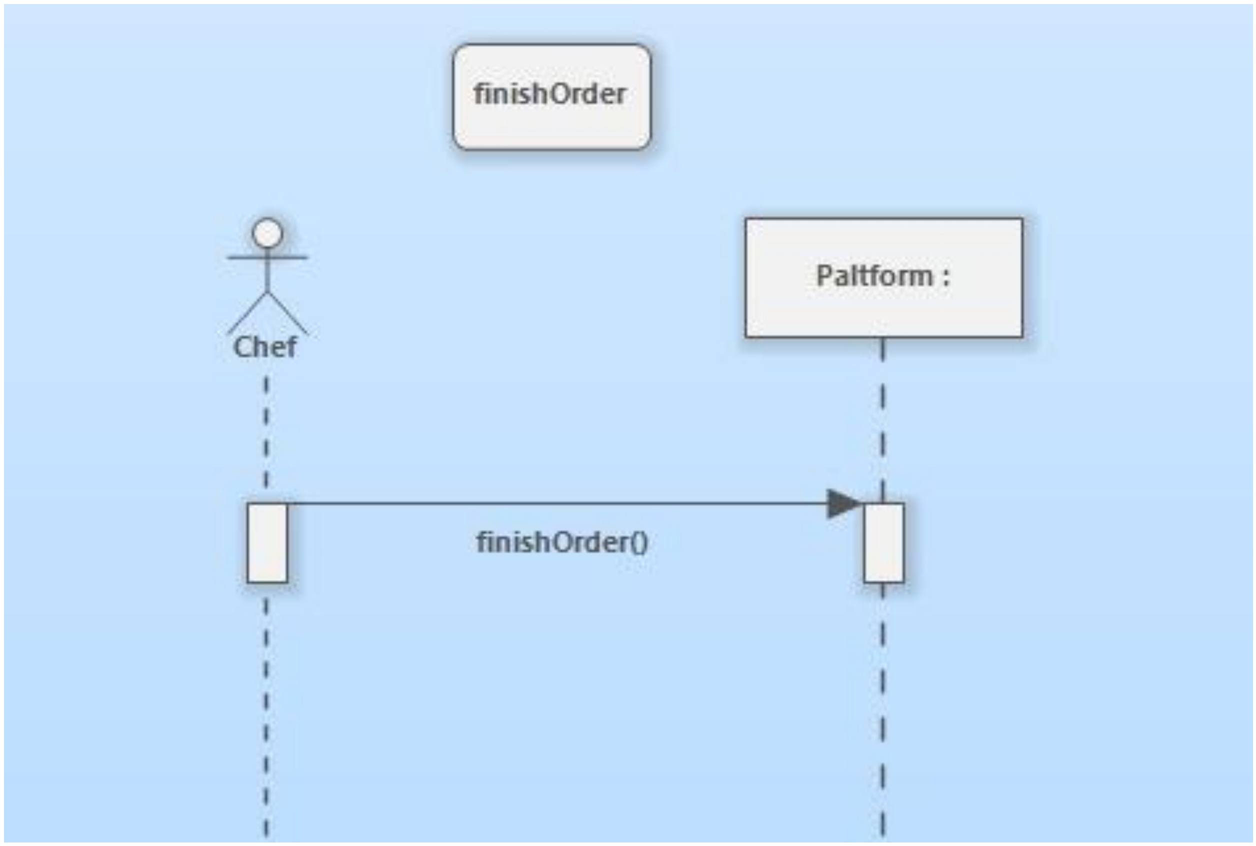
Generate Menu


generate Menu
Manager
Paltform :
Database :
generateMenu(itemName,itemId,item Type,itemCategory,itemPrice,item Description)
Get data
Return
Menu generate

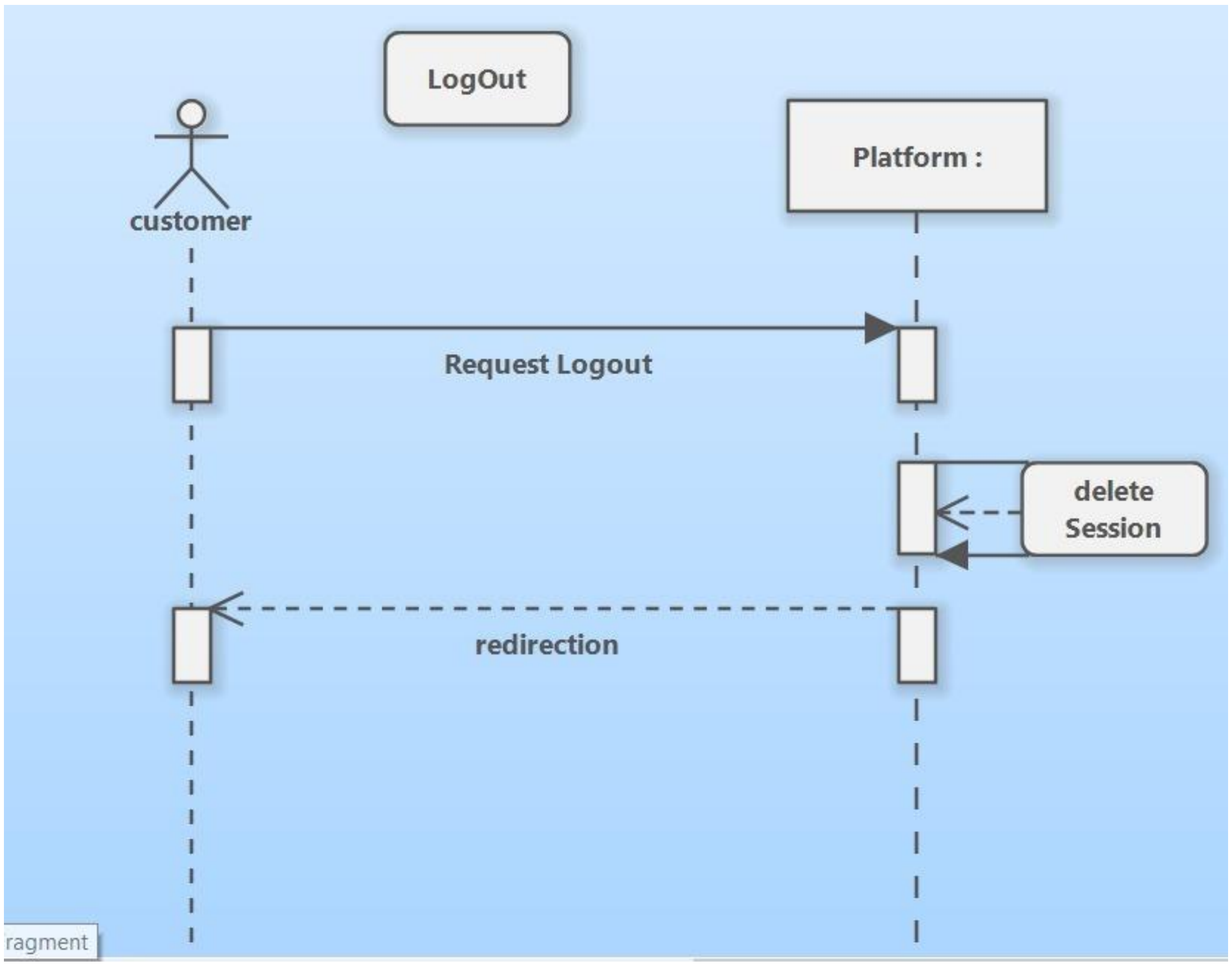
Generate Receipt


generate Recipt
Cahier
Paltform :
Database :
generateRecipt(personName,orderId,orderDate,order Tax,orderPrice,totalAmount,paymentType
Get order data
Return data
Recipt generated

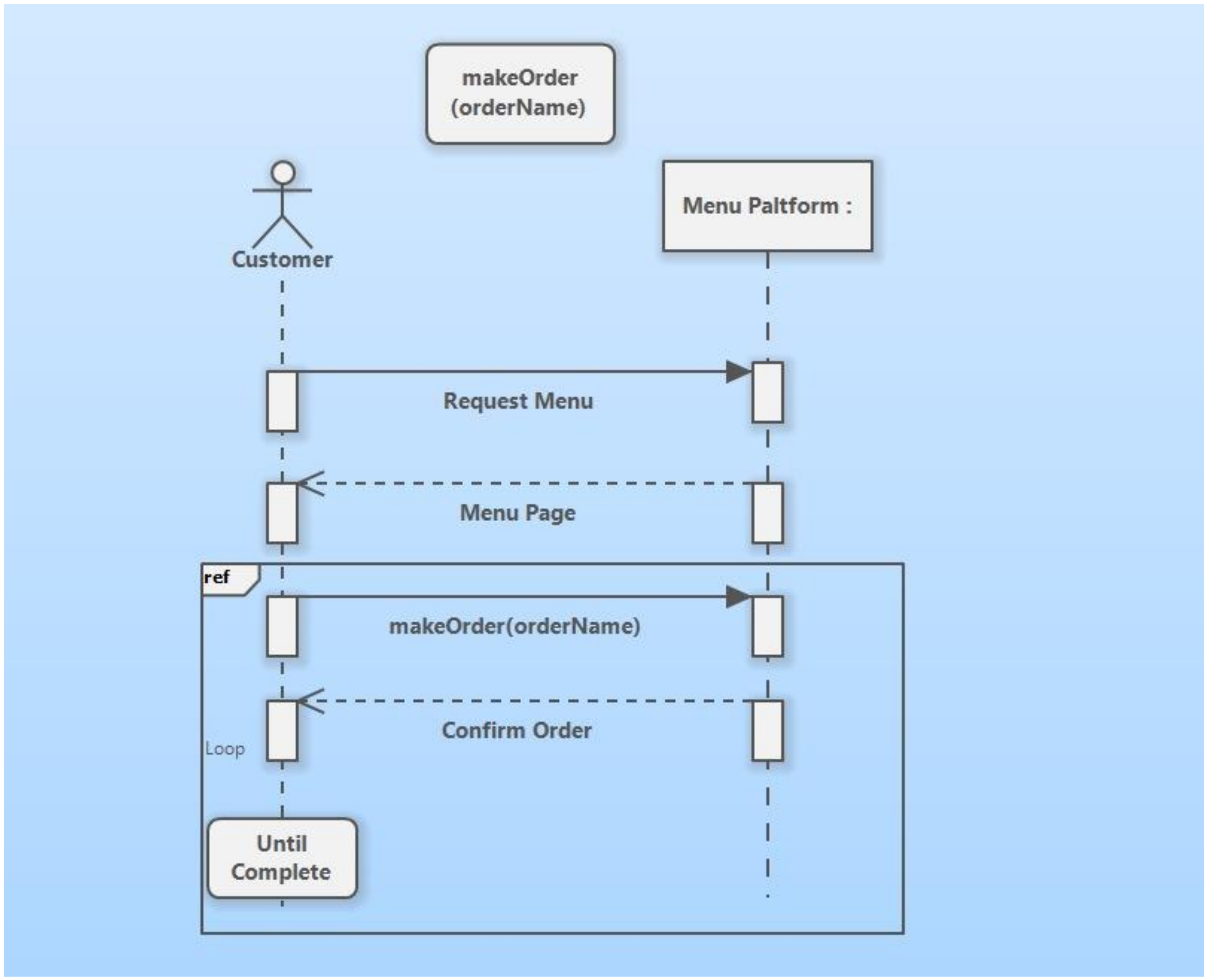
Finish Order



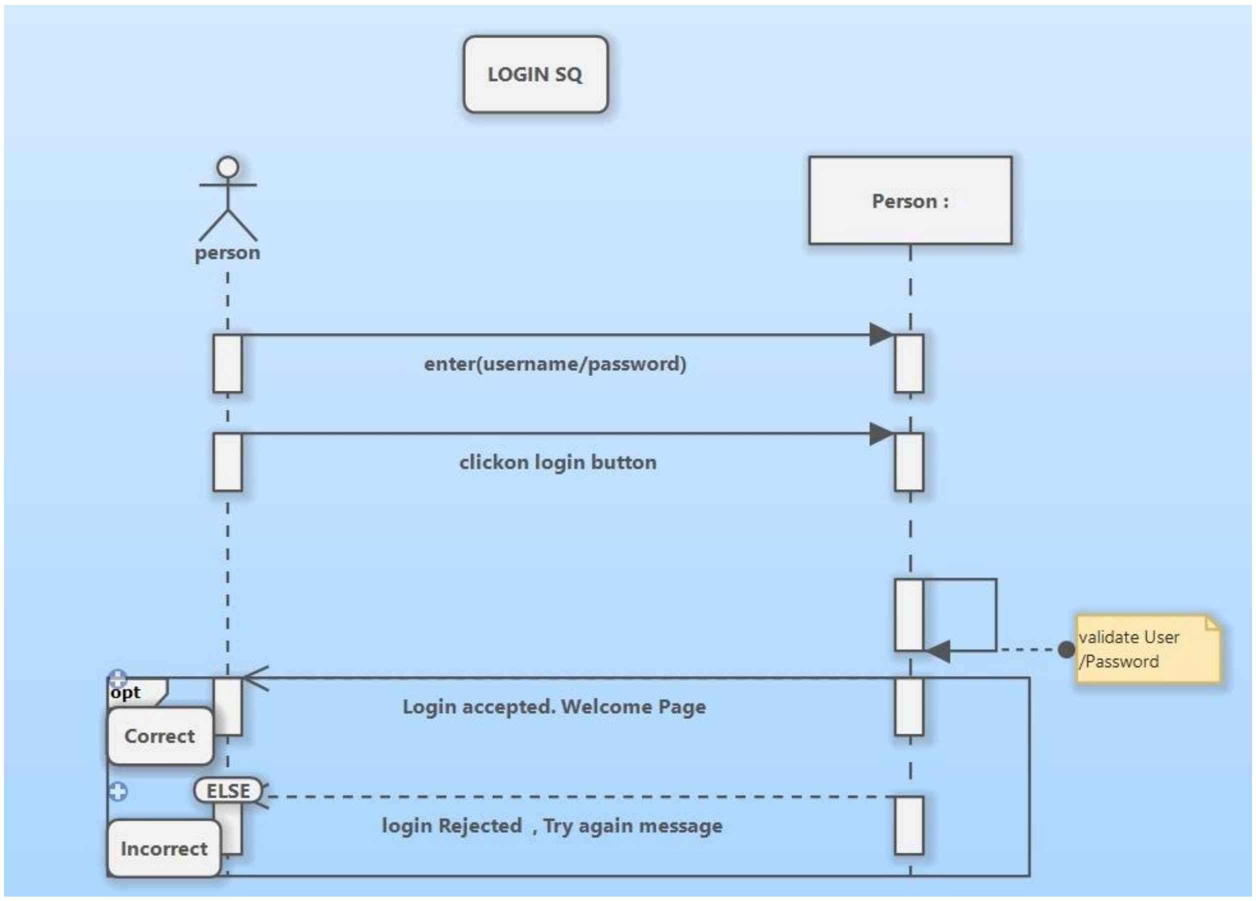
Log Out


LogOut
customer
Platform :
Request Logout
delete Session
redirection
ragment

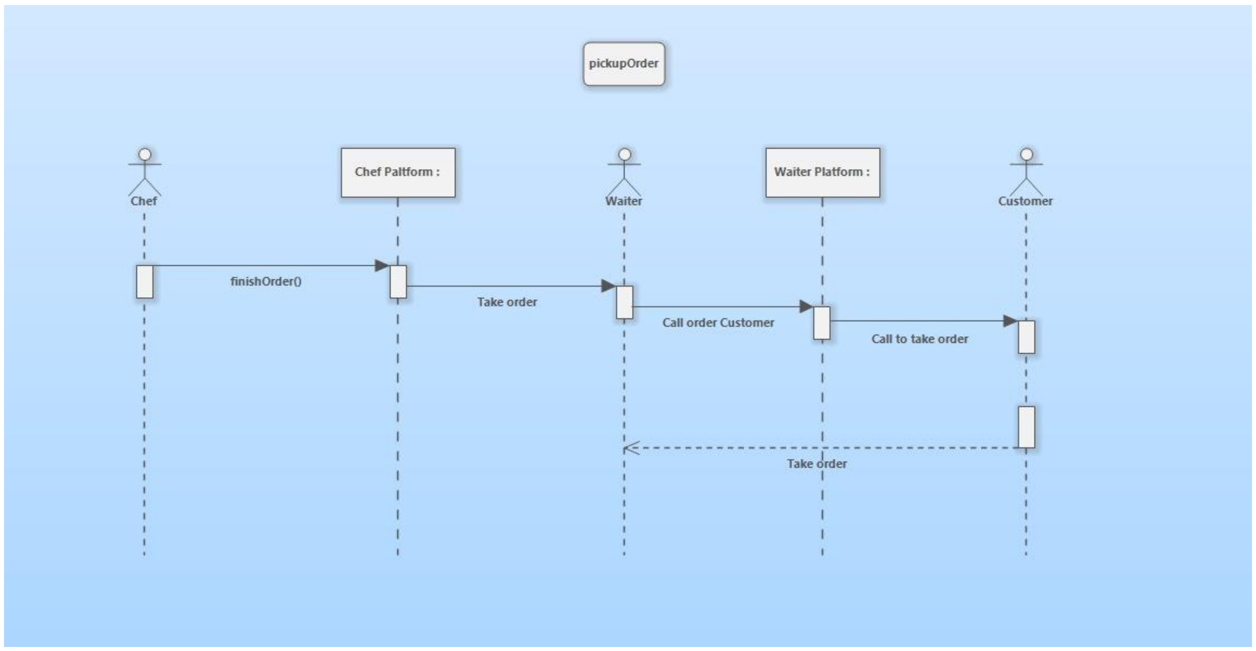
Make Order


makeOrder (orderName)
Customer
Menu Paltform :
Request Menu
Menu Page
ref
makeOrder(orderName)
Confirm Order
Loop
Until Complete

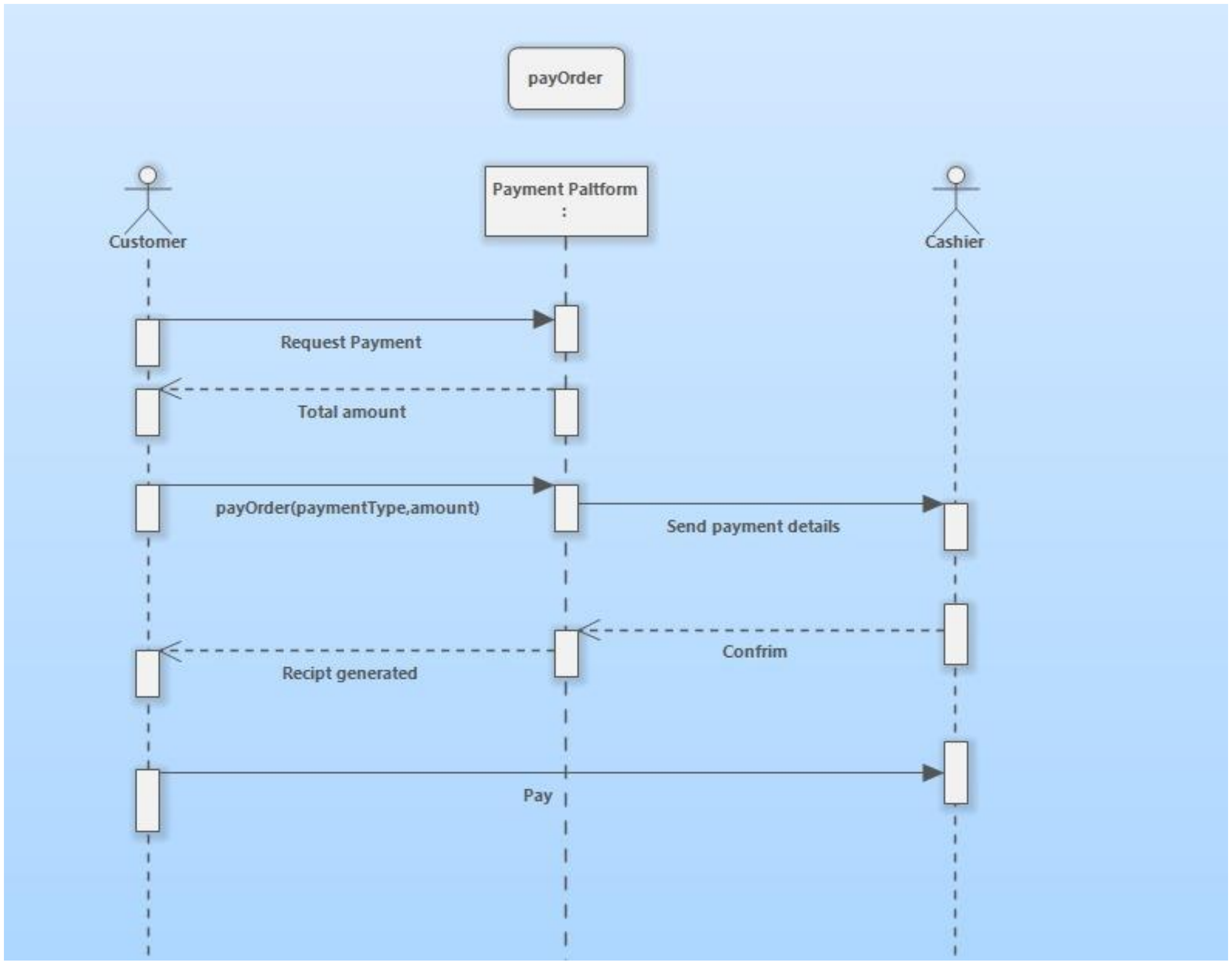
Log In


LOGIN SQ
person
Person :
enter(username/password)
clickon login button
validate User /Password
opt
Login accepted. Welcome Page
Correct
ELSE
login Rejected , Try again message
Incorrect

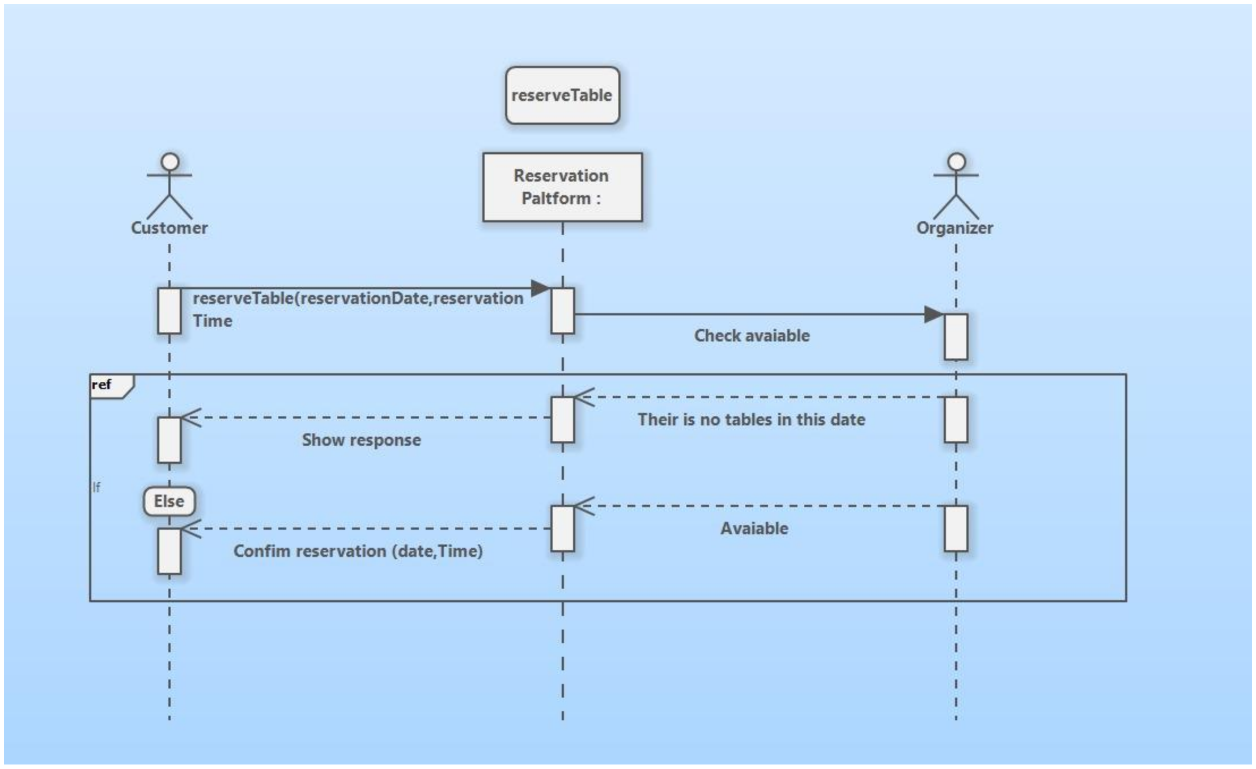
Pick Up Order



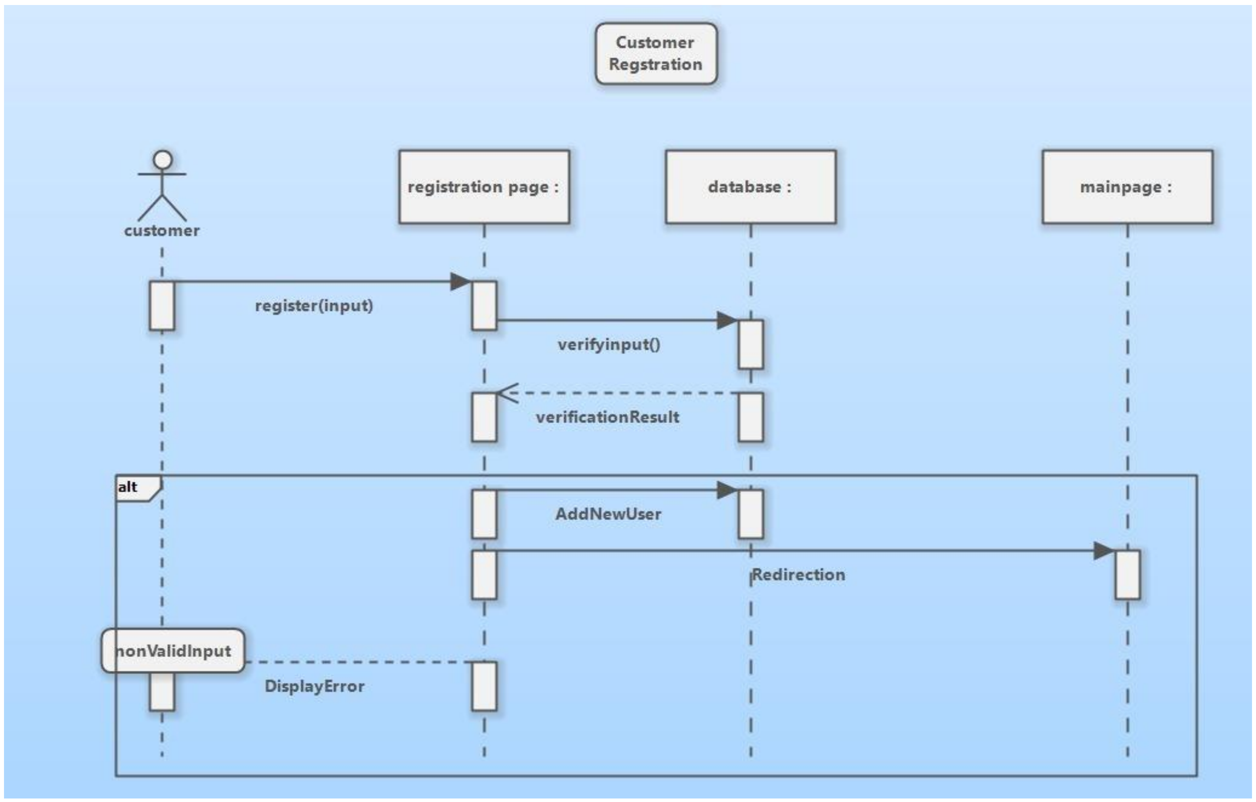
Pay Order


payOrder
Customer
Payment Paltform :
Cashier
Request Payment
Total amount
payOrder(paymentType,amount)
Send payment details
Confrim
Recipt generated
Pay

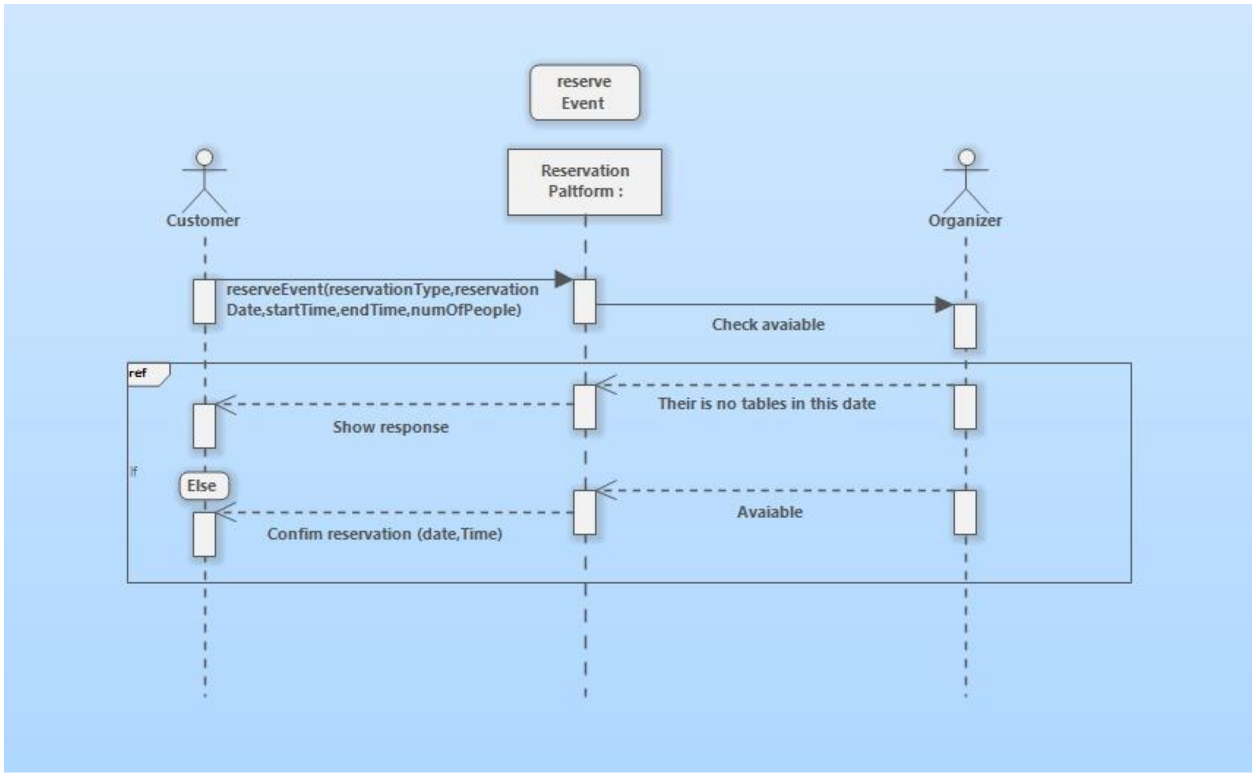
Reserve Table


reserveTable
Reservation Paltform :
Customer
Organizer
reserveTable(reservationDate,reservation Time
Check avaiable
ref
Their is no tables in this date
Show response
If
Else
Avaiable
Confim reservation (date,Time)

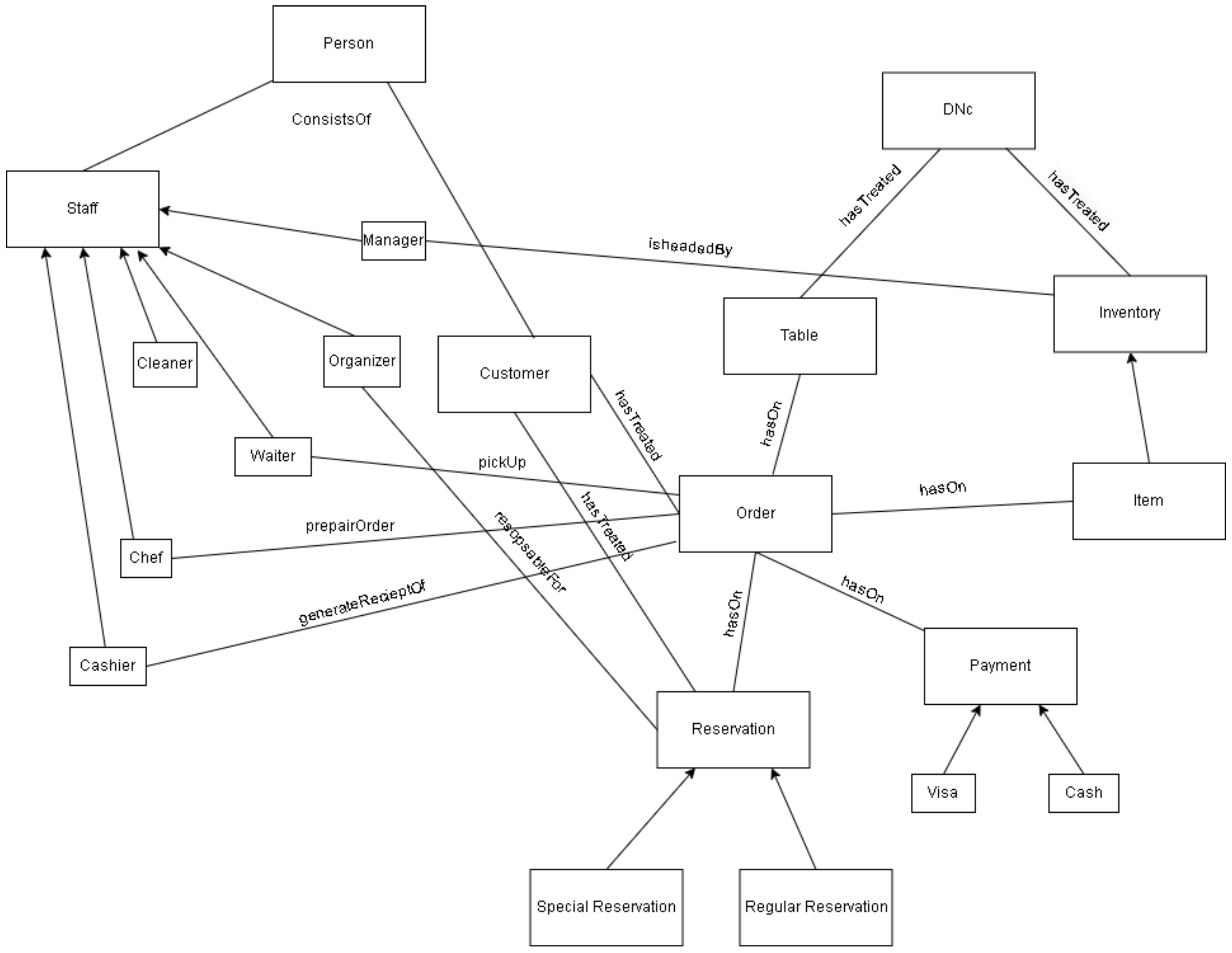
Customer Registration (Sign Up)



Reserve Event


reserve Event
Reservation Paltform :
Customer
Organizer
reserveEvent(reservationType,reservation Date,startTime,endTime,numOfPeople)
Check avaiable
ref
Their is no tables in this date
Show response
If
Else
Avaiable
Confim reservation (date,Time)

Initial Class Diagram



Classes Specifications

Class person	
Instance variable (Attributes)	Comments
Personfirstname…..>string	User enter first name
Personlastname…..>string	User enter last name
PersonAge….>integer	User enter age
personGender…..> string	User select gender
personFullnumber…..>integer	User write full number
personAddress…..>string	User write address
personUsername…>string	User type his username
personPassword…..>string	User type his password
personId…….>integer	User put an id
Instance variables (Associations)	
Customer ----> Staff ----->	
Instance methods	Comments
Signup	An action to register yourself for new account
Login	Security measure designed to prevent unauthorized access to confidential data
Logout	Prevent other users from accessing system without verifying their credentials

Class customer	
Instance variables (Associations)	
order----> Reservation---->	
Instance methods	Comments
Makeorder	Customer order food, drinks, deserts
Cancelorder	Customer can cancel his order before preparations
Payorder	Customer pays for order

Class Manager	
Instance variables (Associations)	
Inventory ---- >	
Staff ---- >	
Instance methods	Comments
Additem	Manager add item on the inventory
Deleteitem	Manager can delete an item from inventory
Searchitem	Manager search items in inventory
Generatemenu	Manager put the menu price and organizes everything on it

Class staff	
Instance variables (Associations)	
Cashier ---> **Waiter--- >** **Organizer---- >** **Manager---- >** **Cleaner---- >**	
Instance methods	**Comments**
Salary	
Education	
Class waiter	
Instance variables (Associations)	
Order ----- >	
Instance methods	**Comments**
Pickup order	When food is prepared the waiter pickup the order to customer
Class chef	
Instance variables (Associations)	
Order ---- >	
Instance methods	
Acceptorder	When the customer make the order the chef accepts it and start to prepare it
Finish order	When chef finishes the order
Class organizer	
Instance variables (Associations)	
Reservation ---- >	
Instance methods	**Comments**
Accept reservation	When customer reserved for an event the organizer accepts reservation
Add plan	Organizer add plan for the event
Class cleaner	
Instance variables (Associations)	
Staff ---- >	
Instance methods	**Comments**
Clean table	When customers finish their food and leave the place the cleaner start to clean the table
Clean floor	When customers finish their food and leave the place the cleaner start to clean the floor
Wash dishes	Cleaner wash dishes
Class cashier	
Instance variables (Associations)	**Comments**
Order ----- >	
Instance methods	
Generatereceipt	When customer finish food and ask for receipt the cashier take the money and generate the receipt

Class reservation	
Instance variables (Associations)	Comments
Regular reservation---- > **Special reservation ---->** **Order---- >** **Customer ---- >**	
Instance methods	Comments
Reservationid	Customer register for event with id
Numberofpeople	Number of people register for event

Class regular reservation	
Instance variables (Associations)	Comments
Reservation ---- >	
Instance variable (Attributes)	
Tableid---- >	Unique id od the table

Class special reservation	
Instance variables (Associations)	Comments
Reservation --- >	
Instance methods	
Reservation type	What is the type of event the customer registers for
Starttime	When it starts
Endtime	When it ends

Class inventory	
Instance variables (Associations)	Comments
DNC---- > **Manager ---- >** **Item**	
Instance variable (Attributes)	Comments
Itemname --- > string	The unique name of item
Itemid---- > integer	The unique id of item
Itemcategory--- > string	
Itemquantity---- > integer	The quantity of item
itemType---- > string	The type of item
Class item	
Instance variable (Attributes)	Comments
Item price--- > double	
Item description---- >string	

Class table	
Instance variables (Associations)	
DNC---- > **Order--- >**	
Instance variable (Attributes)	Comments
Tableid	Unique id for table
Tablecapacity	Size of table
Tablereservation	
Table category	