Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

B. Tech. CE Semester – IV

Subject: Software Project

Project title: Online Food Delivery System

By,

Panchani Nandini P.	
CE082	19CEUEG001

Panchal Vedant V.	
CE081	19CEUBS090

Guided By:-

Prof. Pinkal C. Chauhan

Prof. Brijesh B. Bhatt

Prof. Jigar M. Pandya

Contents:-

1 Abstract:-

The online food ordering system provides convenience for the customers. It overcomes the disadvantages of the traditional queuing system. This system increases the takeaway of foods than visitors. Therefore, this system enhances the speed and standardization of taking the order from the customer. It provides a better communication platform. the user's details are noted electronically

. The online food ordering system set up a menu online and the customers easily place the order with a simple mouse click. Also with a food menu online you can easily track the orders, maintain the customer's database and improve your food delivery service. This system allows the user to select the desired food items from the displayed menu.

The user orders the food items. The payment can be made online. The user's details are maintained confidential because it maintains a separate account for each user. An id and password are provided for each user. Therefore it provides a more secured ordering

2) Introduction:-

In the fast-paced time of today, when everyone is squeezed for time, the majority of people are finicky when it comes to placing a food order. The customers can order online is very convenient but also because they have visibility into the items offered, price, and extremely simplified navigation for the order.

Online food delivery system in which there are Mainly Three users Restaurants, customers and Delivery person. This System is Specially for those Customers Who want to Order Online food In just a minute. Any Restaurants, Customers, or Delivery person Can Register in the System.

Any Restaurants Owner Can log in to the system and add their Restaurants Details and Their Food details with All necessary details. They can Logout of the System too. Another side Any Customer can log in and Order the Food. Costumes can search their Food Types Restaurants vice or a Food type vice. They can add food to the Cart Remove that food also.Customers Add to the Like items and Ordering Next time easily. They Can Payment with Online Methods. Other side Delivery person also log in and delivers the food as usual Restaurant's Owners Commands. Once the order is placed it is entered into the database and retrieved in pretty much real-time.

This allows Restaurant Employees to quickly go through the orders as they are received and process all orders efficiently and effectively with minimal delays and confusion.

3) Software Requirement Specification:-

Online Food Delivery System

1. USER'S ACCOUNT MANAGEMENT:

R.1.1 LOGIN MANAGEMENT:-

Description:The User has to give the necessary login details to log in to the from then they will be redirected to the home page.

Input: User's details

Output: they will be redirected to the homepage.

R.1.2 REGISTRATION MANAGEMENT:-

Description:Users are required to enter the necessary details in the sign-up form to create an account. Then they will create an account by signing up but If Users already have an account they will be redirected to the login page. If not then they will be redirected to the home page.

Input: User's required details

Output: they will be redirected to the home page and get confirmation mail i the registered email account.

R.1.3 EDIT PROFILE:-

Description: In this Users can update their details.

Input: which details the user wants to edit or change they click on it and can update it

Output: A confirmation message.

R.1.3 DELETE PROFILE:-

Description:In this Users can their profile from our website

Input: select option

Output: Redirect to home page

ORDER MANAGEMENT

R.2.1 PLACE ORDER:-

Description:Users can search and place their order bychoosing a restaurant or by choosing any type of food.

Input: select items from search results and move to cart page

Output: cart page.

R.2.2 CANCEL ORDER:-

Description:Users can cancel their order due to any reason.

Input: Reason for canceling the order

Output: confirmation about canceling the order.

3. ADD/DELETE/UPDATE FOOD ITEMS

R.3.1 ADD ITEMS:-

Description: He/She can Add and can delete their food dishes in their account which will be shown on their page.

Input: restaurant's details can be updated

Output: A confirmation message if they will change their profile

R.3.2 DELETE ITEM:-

Input: select item which he/she wants to delete.

Output: deleted item from list.

R.3.3 UPDATE ITEM:-

Input: restaurant's food, it's price or other details which want to update.

Output: update details successfully.

4. PAYMENT MANAGEMENT

R.4.1 PAYMENT:-

Description: users can pay using a different option with payment.

Input: he/she can choose any convenient option for online payment

Output: he/she have to enter account detail and they will get successfull payment.

5. DELIVERY MANAGEMENT

R.5.1 CHANGE STATUS:-

Input: delivery boy has to change the current status to successfully delivered food status.

Output: confirmation message for delivered food.

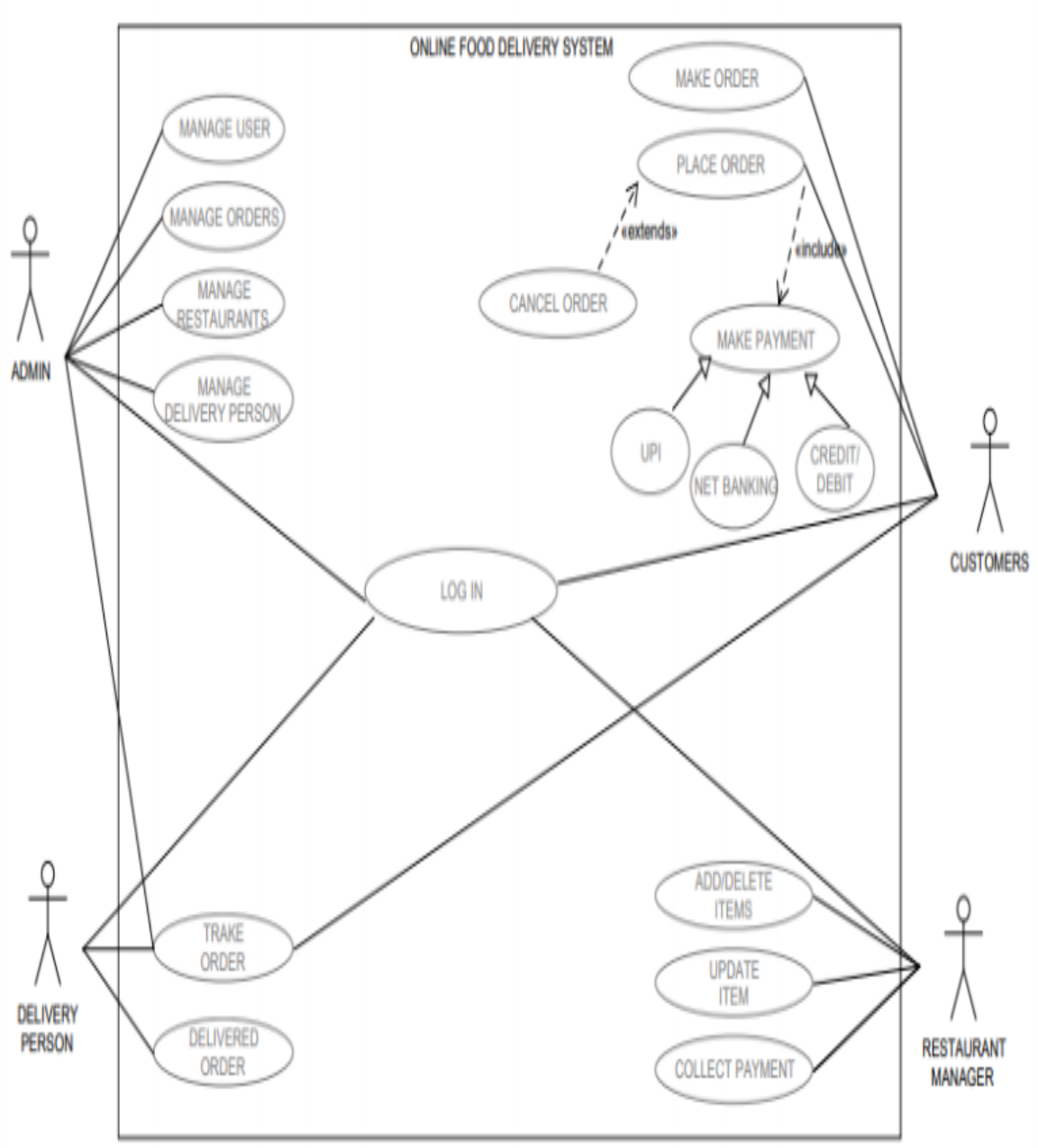
R.5.2 FOOD RATING:-

Input: user will enter the rate of food out of 5 stars.

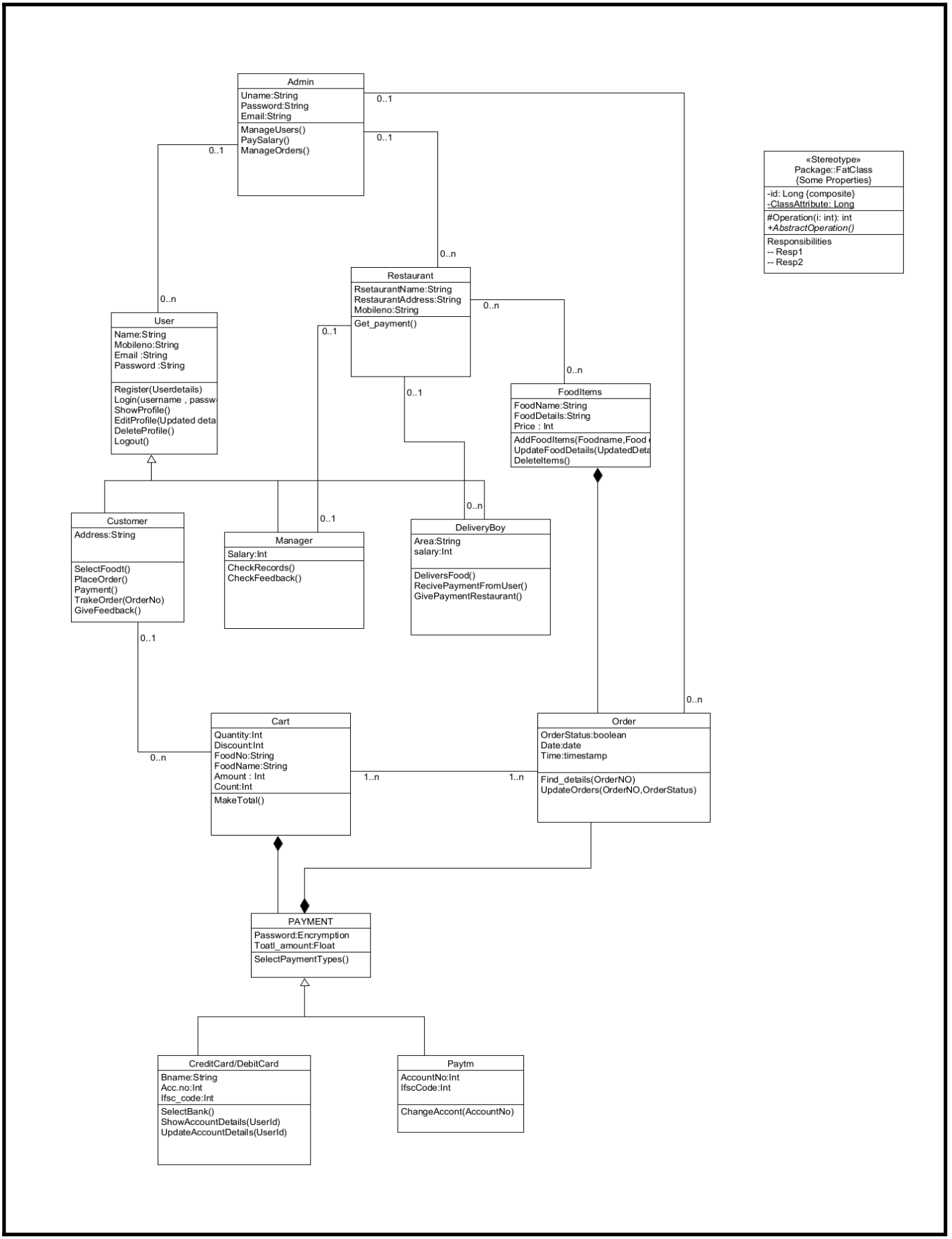
Output: Successfully submitted message

4) Design:

4.1-Use Case diagram:

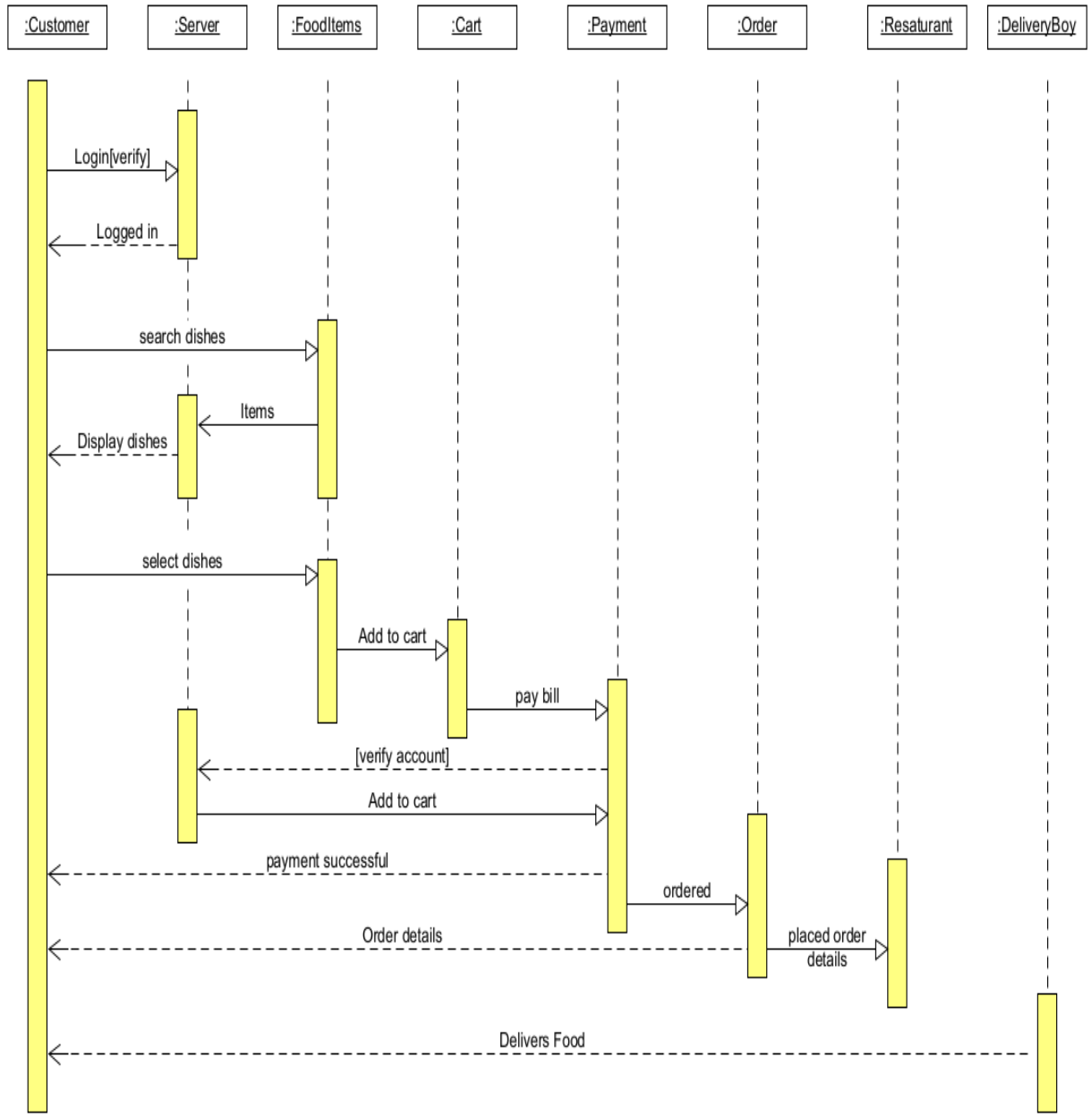


4.2-Class Diagram:

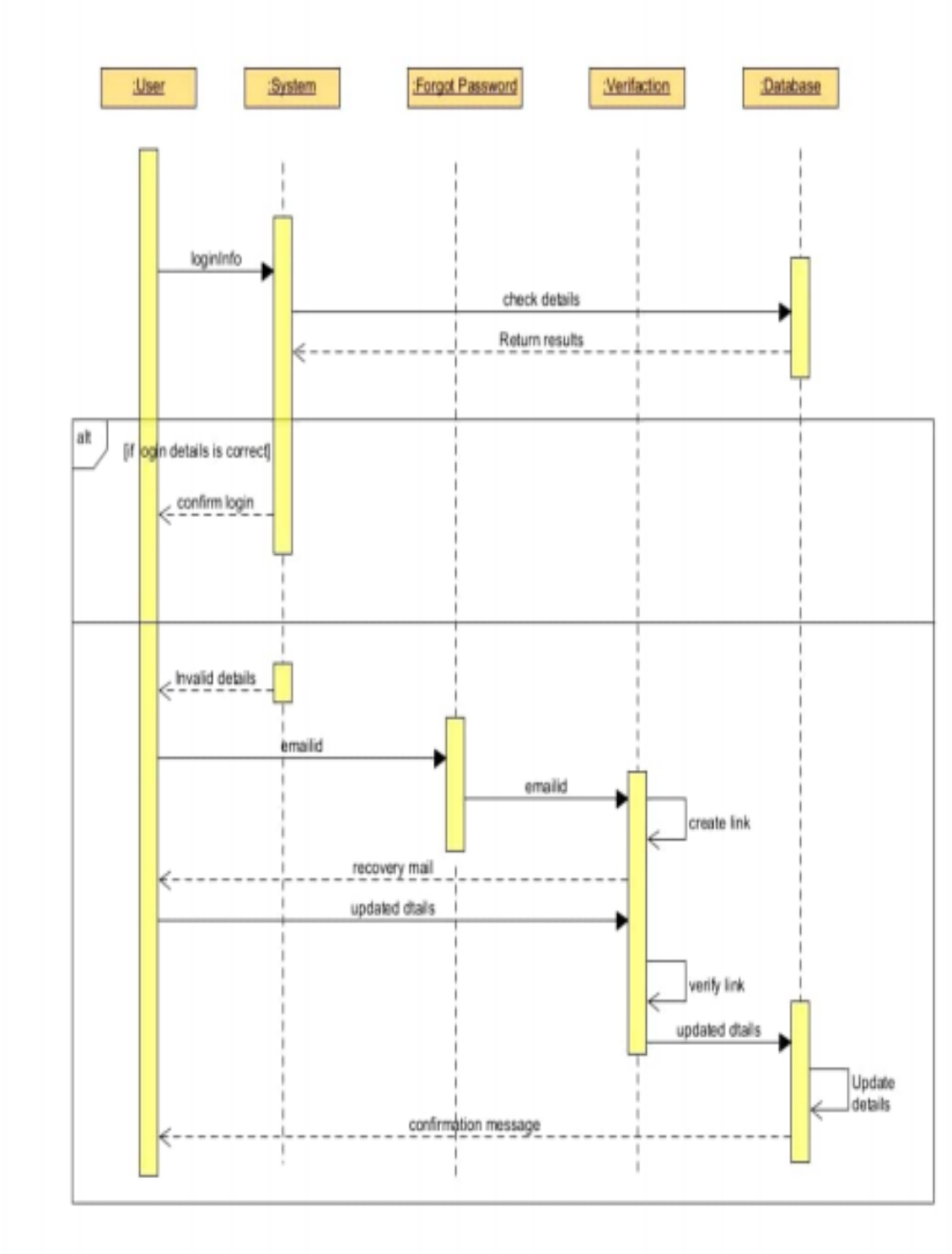

Admin
Uname:String
Password:String
Email:String
ManageUsers()
PaySalary()
ManageOrders()
0..1
0..1
0..1
0..n
«Stereotype»
Package::FatClass
{Some Properties}
-id: Long {composite}
-ClassAttribute: Long
#Operation(i: int): int
+AbstractOperation()
Responsibilities
-- Resp1
-- Resp2
Restaurant
RsetaurantName:String
RestaurantAddress:String
Mobileno:String
Get_payment()
0..n
0..n
User
Name:String
Mobileno:String
Email :String
Password :String
Register(Userdetails)
Login(username , passw
ShowProfile()
EditProfile(Updated deta
DeleteProfile()
Logout()
0..1
0..1
FoodItems
FoodName:String
FoodDetails:String
Price : Int
AddFoodItems(Foodname,Food
UpdateFoodDetails(UpdatedDeta
DeleteItems()
Customer
Address:String
SelectFoodt()
PlaceOrder()
Payment()
TrakeOrder(OrderNo)
GiveFeedback()
Manager
Salary:Int
CheckRecords()
CheckFeedback()
0..1
0..n
DeliveryBoy
Area:String
salary:Int
DeliversFood()
RecivePaymentFromUser()
GivePaymentRestaurant()
0..1
0..n
Cart
Quantity:Int
Discount:Int
FoodNo:String
FoodName:String
Amount : Int
Count:Int
MakeTotal()
0..n
1..n
1..n
Order
OrderStatus:boolean
Date:date
Time:timestamp
Find_details(OrderNO)
UpdateOrders(OrderNO,OrderStatus)
0..n
PAYMENT
Password:Encryption
Toatl_amount:Float
SelectPaymentTypes()
CreditCard/DebitCard
Bname:String
Acc.no:Int
Ifsc_code:Int
SelectBank()
ShowAccountDetails(UserId)
UpdateAccountDetails(UserId)
Paytm
AccountNo:Int
IfscCode:Int
ChangeAccont(AccountNo)

4.3-Sequence Diagram:-

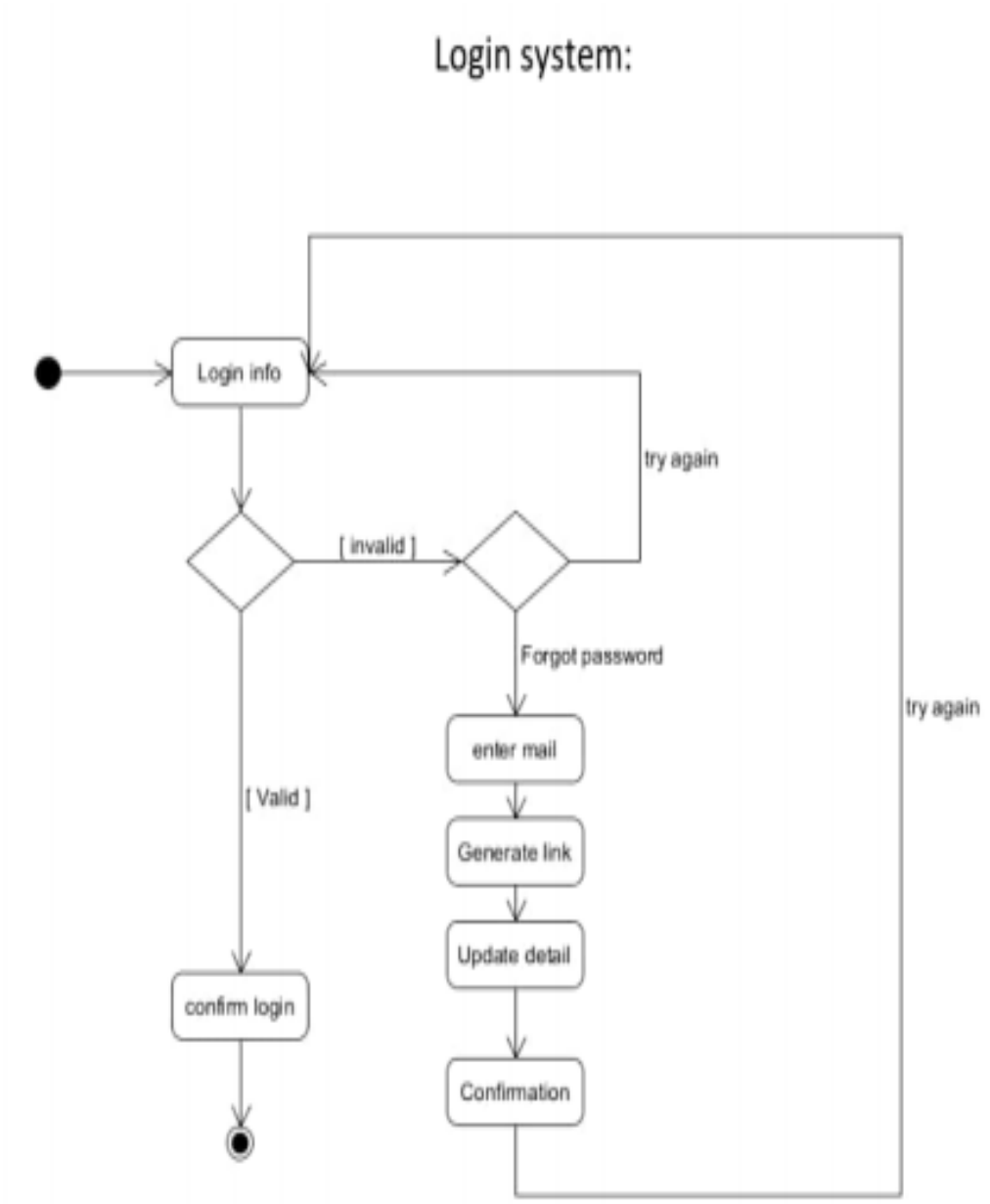
-Order details



-Login System



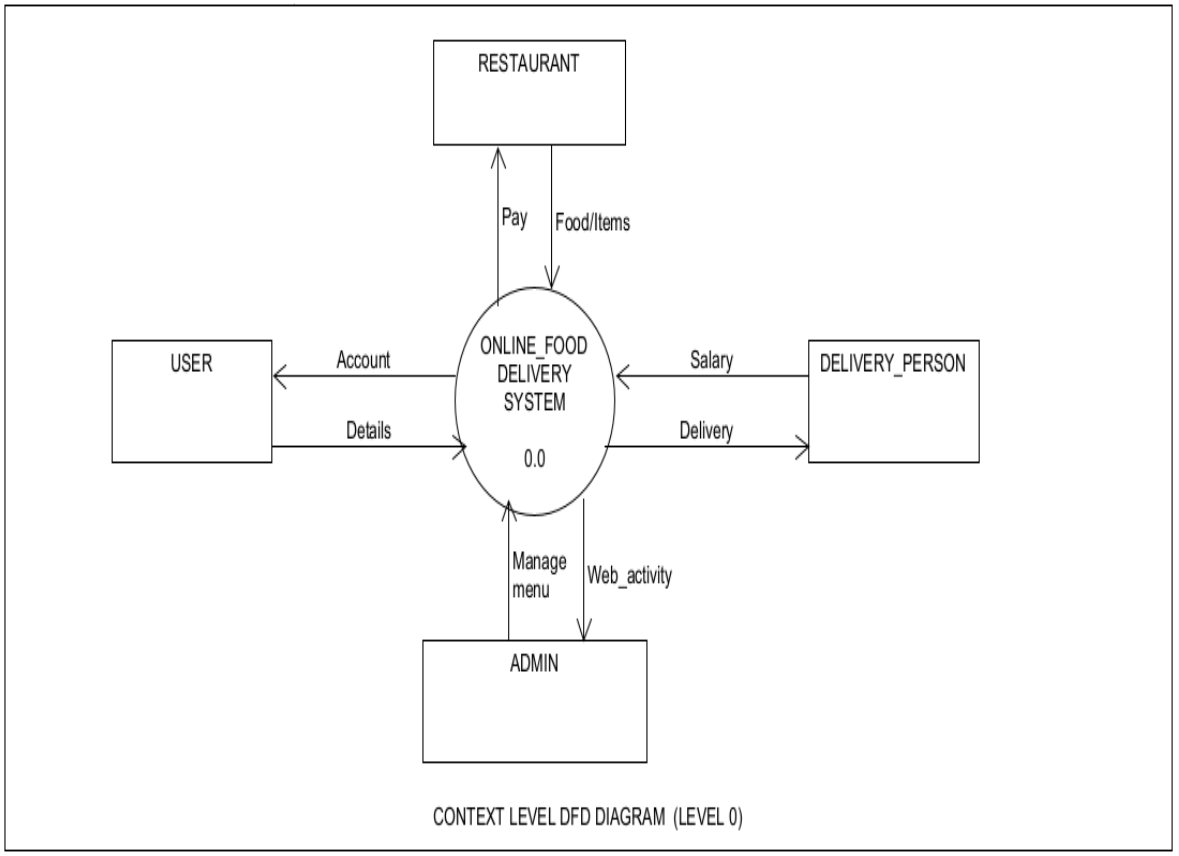
4.4-Activity Diagram:-



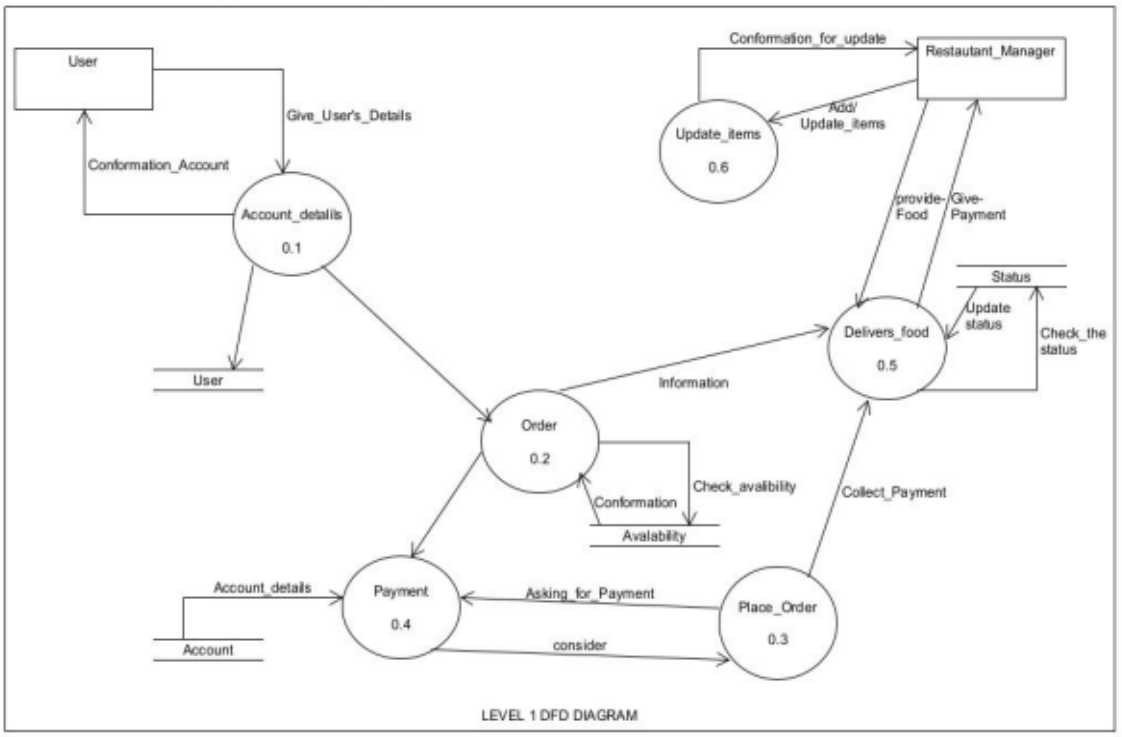

Order System
Loged in
search dishes
[display items]
select dishes
Add to cart
Pay bill
order
Cancle order
Place order
message to resturant
Message to delevery boy
Retum payment
Order details

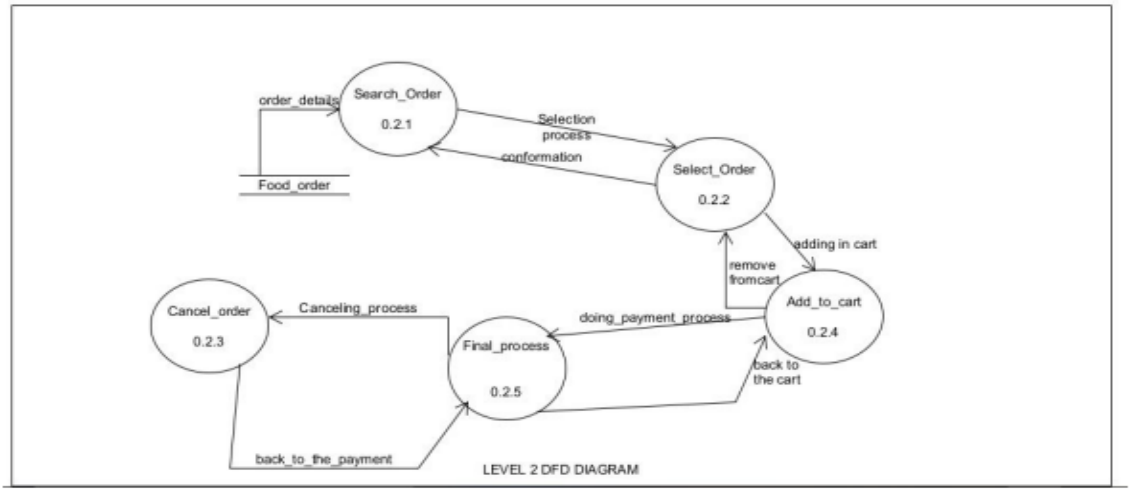
15.

4.5-Data Flow Diagram:



CONTEXT LEVEL DFD DIAGRAM (LEVEL 0)


User
Give_User's_Details
Conformation_Account
Account_details
0.1
User
Conformation_for_update
Restautant_Manager
Add/
Update_items
Update_items
0.6
provide-
Food
Give-
Payment
Status
Update
status
Check_the
status
Delivers_food
0.5
Information
Order
0.2
Check_availibility
Conformation
Availability
Collect_Payment
Account_details
Payment
0.4
Asking_for_Payment
Place_Order
0.3
consider
Account
LEVEL 1 DFD DIAGRAM


order_details
Search_Order
0.2.1
Selection
process
conformation
Food_order
Select_Order
0.2.2
adding in cart
remove
fromcart
Add_to_cart
0.2.4
Cancel_order
0.2.3
Canceling_process
doing_payment_process
Final_process
0.2.5
back to
the cart
back_to_the_payment
LEVEL 2 DFD DIAGRAM

17.

4.6 Data Dictinory:-

User Details

sr. no	Field name	Data type	Width	Unique	PK/FK	Referd Table	Description
1	id	Int	11	yes	yes	-User account -cart -order product	Auto Increment
2	Username	varchar	20	yes	no	-	
3	Email	varchar	50	yes	no	-	
4	Mobile no.	varchar	10	yes	no	-	
5	Password	varchar	20	no	no	-	
6	Address	varchar	150	no	no	-	
7	State	varchar	20	no	no	-	
8	City	varchar	20	no	no	-	
9							

User account details

Sr. no	Field name	Data type	Width	Unique	PK/FK	Refered Table	Description
1	Id	Int	11	yes	yes	-user details	Auto Increment
2	Account No	Int	12	yes	no	-	
3	Bank Pass.	varchar	12	no	no	-	
4	User Account id	Int	11	yes	no	-	

Restaurant Details

Sr. no	Field Name	Data type	Width	Unique	PK/FK	Refered table	Description
1	Id	Int	11	yes	yes	-restaurant account	Auto Increment
2	Restaurant name	varchar	50	no	no	-	
3	Restaurant Photo	varchar	100	no	no	-	
4	Status	tinyny	1	no	no	-	
5	Password	varchar	30	no	no	-	
6	Username	varchar	30	yes	no	-	
7	Email	varchar	50	yes	no	-	
8	Mobile no.	varchar	10	yes	no	-	
9	Address	varchar	150	yes	no	-	
10	state	varchar	20	no	no	-	
11	City	varchar	20	no	no	-	
12	Feedback	decimal	1,1	no	no	-	

Cart

Sr. no	Field Name	Data type	Width	Unique	PK/Fk	Refered Tablr	Description
1	Id	int	11	yes	yes	-user details	Auto Incerement
2	Quantity	Int	11	no	no	-	
3	Final Amount	decimal	19,10	no	no	-	
4	Items data id	Int	11	yes	yes	-	
5	User data id	Int	11	yes	yes	-	

Restaurant items

Sr. no	Field Name	Data type	Width	Unique	PK/FK	Refered table	Description
1	Id	Int	11	yes	yes	-restaurant details	Auto Increment
2	Name	varchar	50	no	no	-	
3	desc	varchar	150	no	no	-	
4	price	Decimal	19,10	no	no	-	
5	Image	varchar	100	yes	no	-	
6	Category	varchar	150	no	no	-	
7	Amount	Decimal	19,10	no	no	-	
8	Discount	Decimal	19,10	no	no	-	
9	Feedback	Decimal	19,10	no	no	-	

Order Product

Sr. no	Field Name	Data type	Width	Unique	PK/FK	Refered table	Description
1	Id	Int	11	yes	yes	-cart -user details	Auto Increment
2	Date	datetime	6	no	no	-	
3	Status	varchar	150	no	no	-	
4	Price	Decimal	19,10	no	no	-	
5	Cart id	Int	11	yes	yes	-	
6	User id	Int	11	yes	yes	-	

Delivery boy Details

sr. no	Field name	Data type	Width	Unique	PK/FK	Referd Table	Description
1	id	Int	11	yes	yes	-order product	Auto Increment
2	Username	varchar	20	yes	no	-	
3	Email	varchar	50	yes	no	-	
4	Mobile no.	varchar	10	yes	no	-	
5	Password	varchar	20	no	no	-	
6	Address	varchar	150	no	no	-	
7	State	varchar	20	no	no	-	
8	City	varchar	20	no	no	-	
9							

Restaurant account

Sr. no	Field name	Data type	Width	Unique	PK/FK	Refered Table	Description
1	Id	Int	11	yes	yes	-	Auto Increment
2	Account No	Int	12	yes	no	-	
3	Bank Pass.	varchar	12	no	no	-	
4	User Account id	Int	11	yes	no	-	

5 Implementation Details:-

5.1 Module:-

The system consists of 4 basic modules namely

1. User Module
2. Restaurant Module
3. Delivery Boy Module
4. Order Module

Each module consists of several methods to implement the required functionality. Implementation is done using Django. Database used in these modules is MySQL.

1.User Module:-

This module is the base for authentication and authorization to ensure the security aspect of the user. It also includes profile creation,login,logout,forgotpassword and update it if required.

2.Restaurant Module:-

This module handles the functions related to Restaurants. It allows Restuartant's owner to login,logout and update the details of Restaurants as well as Food details. It also displays Restaurants name and items at tho home page.

3.Delivery Boy Module:-

This module is the base for authentication and authorization to ensure the security aspect of the user. It also includes profile creation,login,logout,forgotpassword and update it if required. And it delivered the food as ordered of restaurants.

4.Order Module:-

This module handles various Orders of the customers. It also handles the cart and payment method also. User cancel the order by this module.

5.2 Function prototypes:-

Login

```
def login(request):
    context={}
    if request.method=="POST":
       username=request.POST['email']
       password=request.POST['password']
       user = auth.authenticate(username=username,password=password)
       if user is not None:
            if customer.objects.filter(uname=username).exists():
                user1=customer.objects.get(uname=username)
                print("yes")
                if user1 is not None:
                    auth.login(request,user)
                    request.session['customer_id']=user1.id
                    request.session['user_id']=user.id
                    return render(request,"OrderingOnline/index.html")
                else:
                    context['not_exist'] ="User is not exist"
                    return render(request,'OrderingOnline/login.html',context)
            else:
                context['not_exist'] ="User is not exist"
                return render(request,'OrderingOnline/login.html',context)
    return render(request,'OrderingOnline/login.html',context)


def signup(request):
    if request.method=="POST":
```

Add to cart-

```
#adding data in cart
@login_required(login_url='/OrderingOnline/login')
def add_to_cart(request,rest_id,id,page):
    context={}
    context['cart_count']=cart.objects.all().count
    Item_data=Item.objects.get(id=id)
    print(Item_data.id)
    cust=customer.objects.get(id=request.session['customer_id'])
    print(cust.id)
    check_data=cart.objects.filter(userdata=cust,Items_data=Item_data)
    print(check_data)
    quantity=1
    if check_data:
        try:
            if page == 'show_restaurant_items':
                messages.info(request,Item_data)
                messages.warning(request, "You have already this dish in cart")
                return HttpResponseRedirect("/OrderingOnline/show_restaurant/{}".format(rest_id))
            if page == 'show_dish':
                messages.info(request,Item_data)
                messages.warning(request, "You have already this dish in cart")
                return HttpResponseRedirect("/OrderingOnline/showdetails/{0},{1}".format(id,rest_id))
        except:
            return HttpResponseRedirect("/OrderingOnline/showdishes/{0}".format(id))
    else:
        final_amount=(float(Item_data.price)-float(Item_data.discount))
        print(final_amount)
        final_amount=float(quantity)*(final_amount)
```

Place Order

```
@login_required(login_url='/OrderingOnline/login')
def placeorder(request):
    context={}
    context['cart_count']=cart.objects.all().count
    user=customer.objects.get(id=request.session['customer_id'])
    cart_data=cart.objects.filter(userdata=user)
    price=0.0
    for x in cart_data:
        price+=x.quantity*(float(x.Items_data.price)-float(x.Items_data.discount))
        print(x.Items_data.pname)
    context['price']=price
    context['cart']=cart_data
    context['confirm']="Do You want to order? "
    if request.method=='POST':
        context['confirm']=''
        for x in cart_data:          # for fetching data from order
            # item=Item.objects.get(id=x.Items_data_id)
            order_place=OrderProduct(price=x.final_amount,userdata=user,cartdata=x)
            order_place.save()
        print(price)
        order=OrderProduct.objects.filter(userdata=user)
        print("order-total ",price)
        context['order']=order
        context['total']=price
        return render(request,"OrderingOnline/confirmation_order.html",context)
    return render(request,"OrderingOnline/cart.html",context)
```

Payment(customer Side)

```
@login_required(login_url='/OrderingOnline/login')
def payment(request):
    context={}
    if request.method=="POST":
        uname=request.POST['uname']
        account=request.POST['account_no']
        print(uname,account)
        try:
            user_data=customer.objects.get(uname=uname)
            try:
                user=Account_user.objects.get(user_account=user_data)
                if user:
                    account_user=Account_user.objects.get(account_no=account)
                    print(account_user)
                    otp=save_otp(user_data.mobile)
                    request.session['otp']=otp
                    # print("s1",otp)
                    context['otp']=otp
                else:
                    context['detail']="Account no is wrong"
            except Account_user.DoesNotExist:
                print("no")
                context['not_saved']=['Not saved']
        except:
            context['detail']="Account user is not exist"
    return render(request,'OrderingOnline/payment.html',context)
```

Add Items (Restaurants Side)

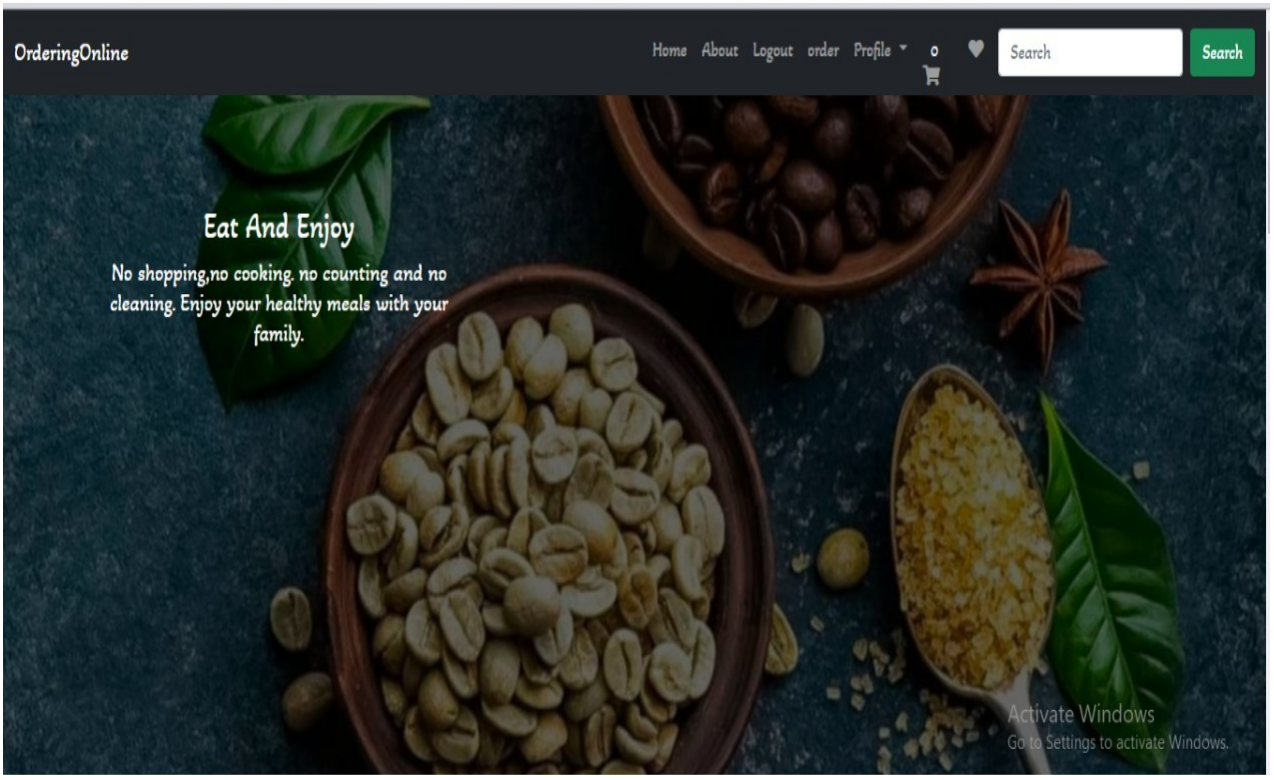
```
#add item in resaturant
@login_required(login_url='/restaurant/r_login/')
def add_items(request):
    context={}
    cat=Category.objects.all()
    context['cat']=cat
    if request.method=="POST":
        rdata=restaurant.objects.get(uname=request.session['r_user'])
        print(rdata)
        food=request.POST['foodname']
        category=request.POST.getlist('category',None)
        desc=request.POST['desc']
        price=request.POST['price']
        image=request.FILES['foodphoto']
        if filetype.is_image(image):
            data_item=Item(pname=food,pdesc=desc,price=price,pimage=image)
            data_item.save()
            data_item.rdata.add(rdata)
            context['success']="Successfully added"
        else:
            context['error']="Please enter Image file"
    return render(request,"Restaurant1/additems.html",context)
```

Show items restaurants vice and food type vice-

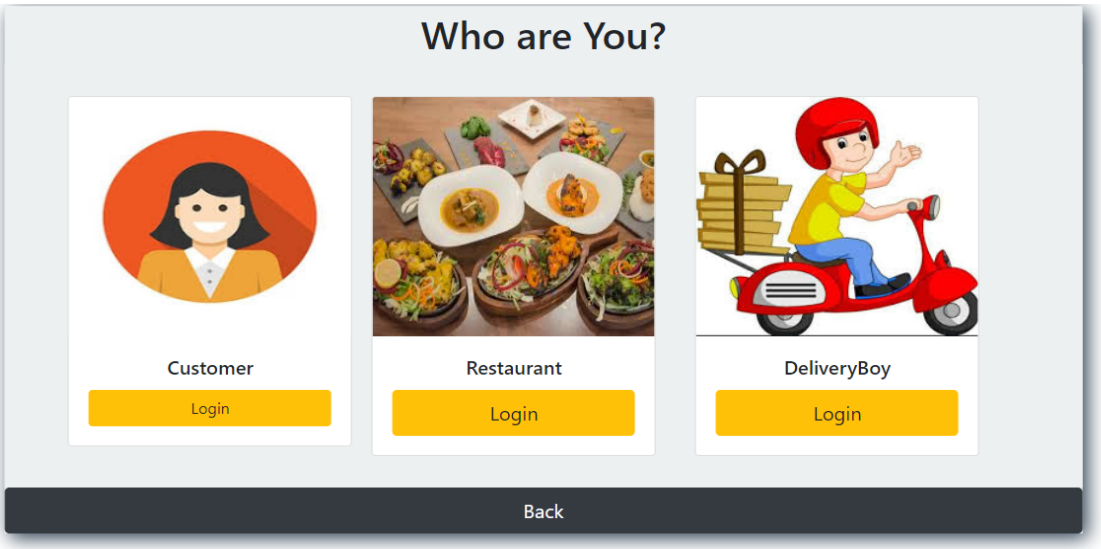
```
# food dish show
def show_dish(request,id,rest_id):
    context={}
    context['cart_count']=cart.objects.all().count
    Item_data=Item.objects.get(id=id)
    context['Item_data']=Item_data
    context['rest_id']=rest_id
    return render(request,"OrderingOnline/show_dish.html",context)


# show restaurnat items
def show_restaurant(request,id):
    context={}
    context['cart_count']=cart.objects.all().count
    items=Item.objects.filter(rdata=id)
    context['items']=items
    context['rest_id']=id
    return render(request,"OrderingOnline/show_restaurant_items.html",context)
```

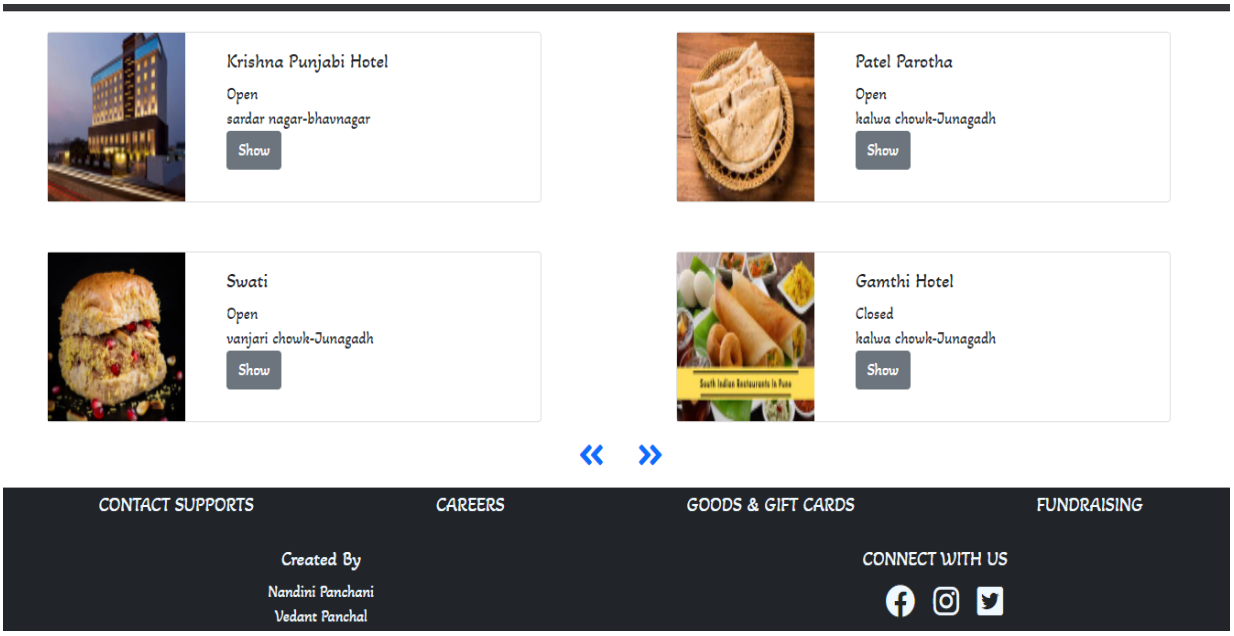
6) ScreenShots:-



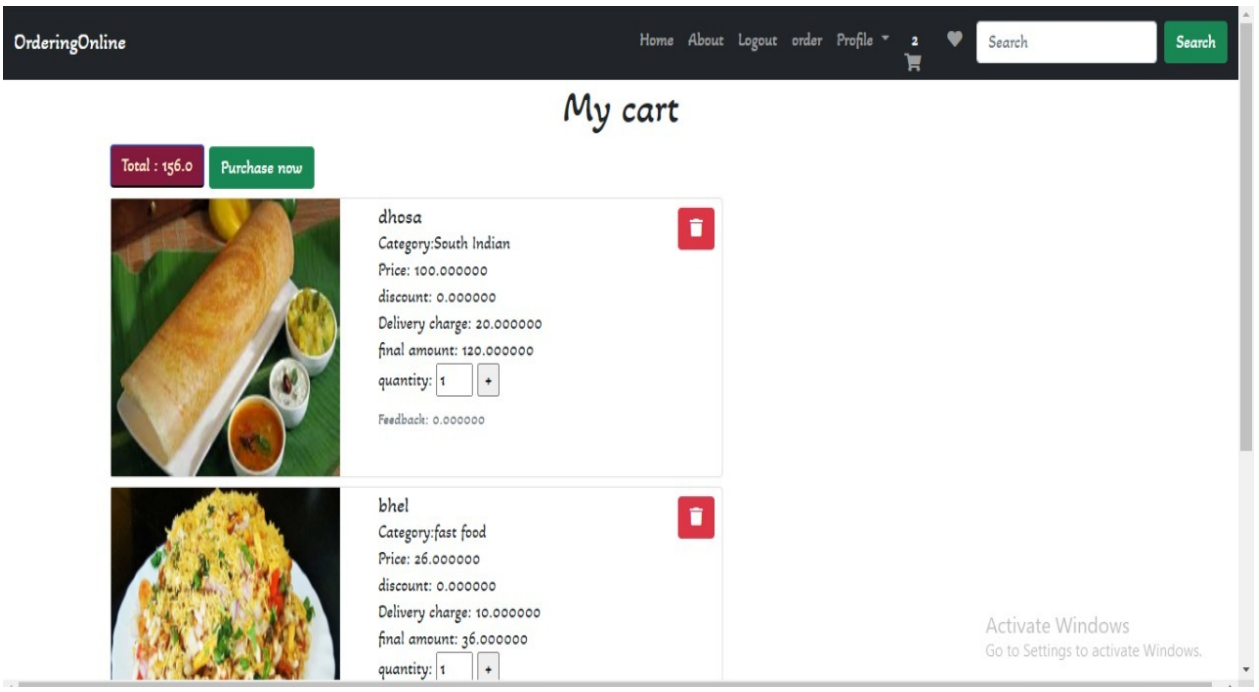
Home Page



Login Page

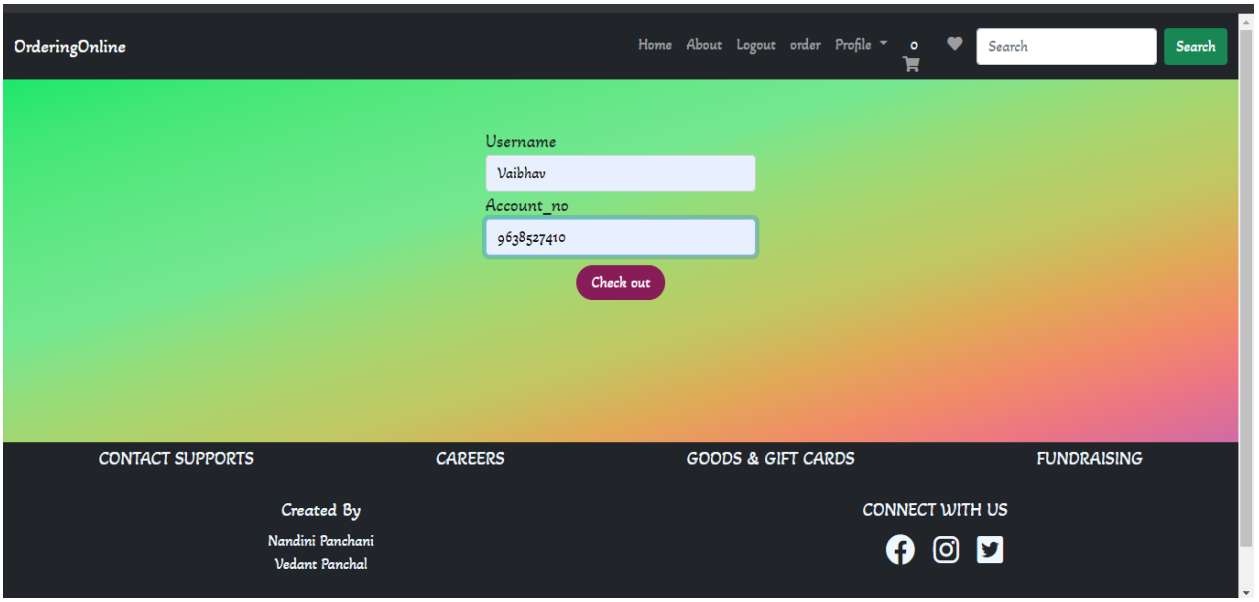


Restaurant Details for Ordering Food

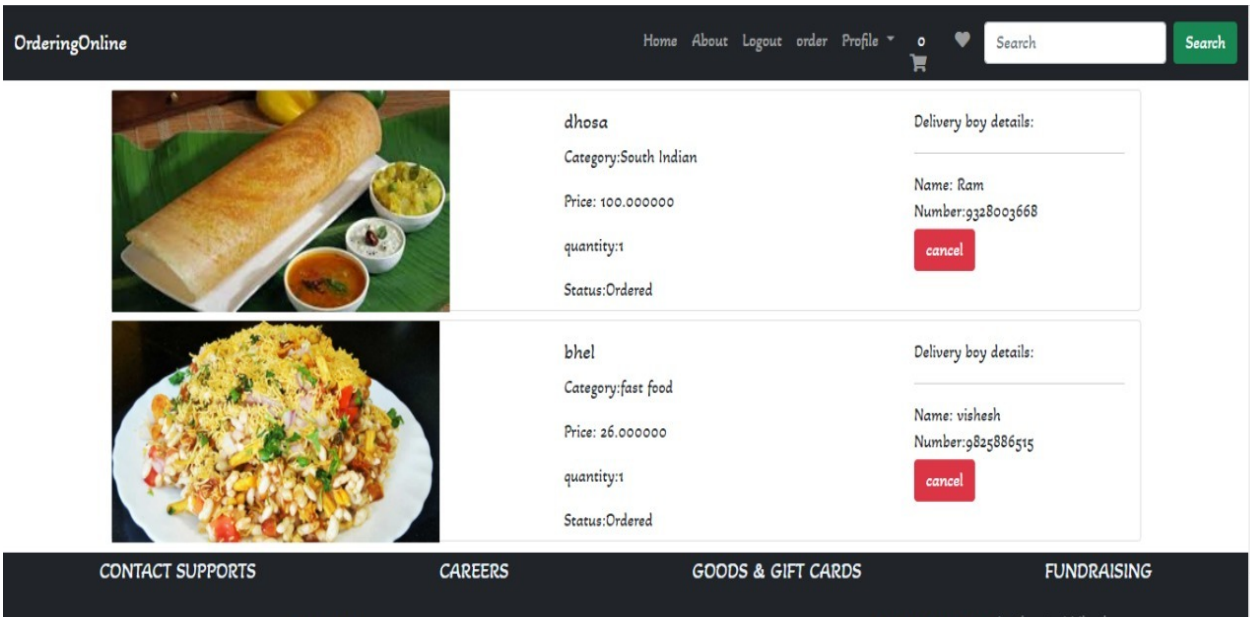


Cart **28**

Restaurant Side Details



Payment Details



Final Order With Delivery Person Details

Delivery boy Details

7) Conclusion:-

The functionalities are implemented in system after understanding all the system modules according to the requirements. Functionalities that are successfully implemented in the system are:

- User,Restaurant and Delivery person registration containing all the necessary validation on field
- Login
- authentication
- Logout
- Forgot Password
- Update/Delete Profile.
- Add/Update/Delete items (Restaurant side)
- Add/Delete items from cart
- showed and like items
- place order/ cancel order
- payment(dummy)
- Track order

After the implementation the coding of system, user interface and security issues related to system And Finally the system is implemented.

8) Limitations and Future Enhancements:-

Limitations:-

- We are unable to do Real Payment through bank account so we created dummy account for it.
- User can not Order different restaurants at the Same time with Same cart thats our expectations but in our system user can do it so thats our limitation.

Future Enhancements:-

- User can also give for restaurant<----restaurant feedback---->
- User only add to cart if dish is same from resaturant which cart
- payment gateway(Real)
- maintain all waiting dishes automatically
- UI designs.
- contact us via mail system
- Email verification after signup
- filters
- offer codes uses at payment
- If delivery boy get more than one orders and also othen has no order even they haven't order beacause of deliveryboy who is available but got many times job is first available in database (delivery boy selecion update).
- Manual Admin panel

9) **Reference / Bibliography:-**

Following links and websites were referred during the development of this project:

Stackoverflow.com

Github.com

Docs.djangoproject.com

Bootstarp.com

Thank You