A Project Report on

Restaurant Management System

By

Parmar Darshan Kiritbhai

(CE084) (19CEUBG023)

Patel Harshkumar Nayanbhai

(CE098) (19CEUON066)

B. Tech

CE Semester-IV

Subject: SP + SEPP

Guided by:

Prof. Jigar M. Pandya,

Prof. Pinkal C. Chauhan,

Prof. Brijesh S. Bhatt

Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

DHARMSINH DESAI UNIVERSITY

NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled as "Restaurant Management System" is a bona-fide report of the work carried out by

- **Parmar Darshan Kiritbhai,** Student Id: **19CEUBG023**
- **Patel Harshkumar Nayanbhai,** Student Id: **19CEUON066**

of Department of Computer Engineering, semester **IV**, under the guidance and supervision of **Prof. Jigar M. Pandya** for the subject **Software Engg. & Practices** during the academic year 2020-2021.

Project Guide
Assistance Professor
Prof. Jigar M. Pandya
Department of Computer Engineering,
Faculty of Technology,
Dharmsinh Desai University,
Nadiad

Head of the Department
Prof. & Head,
Dr. C.K Bhensdadia
Department of Computer Engineering,
Faculty of Technology,
Dharmsinh Desai University,
Nadiad

Contents:

Abstract:

Online Food Delivery System is a system which will help restaurant to optimized and control over their restaurants. For the waiters, it is making life easier because they don't have to go kitchen and give the orders to chef easily. For the management point of view, the manager will able to control the restaurant by having all the reports to hand and able to see the records of each employees and orders. This website helps the restaurants to do all functionalities more accurately and enhances the spend of all the tasks. Online Food Delivery System reduces manual works and improves efficiency of restaurant. The online food delivery system set up menu online and the customers easily places the order with a simple mouse click. Also with a food menu online you can easily track the orders, maintain 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 or pay-on-delivery system. The user's details are maintained confidential because it maintains a separate account for each user. An id and password is provided for each user. Therefore, it provides a more secured ordering.

Introduction:

Brief Introduction:

Smart Restaurant Management System is a new generation of restaurant management software. When users/customer will enter in the website, he/she should have an account. If user does not have an account, user has to create a new account to order food. To create a new account user should enter unique username, email and new mobile no. with password. User fill his/her address for food delivery. Once user enters in the website, you can see different types of food available in restaurant. First select category of food from soups, starters, the main course dishes and desserts. After that search food as your interest. select food you want to order. After selecting all your meal place your order and confirm your address. Then website will saw you various type of payment methods and your total bill amount. You can pay cash on delivery or there are many more options for online payment to get benefits. online payment methods:

- Credit/Debit card payments
- Bank transfers
- E-Wallets
- UPI payments

You can choose best deal for your meal.

Tools/Technologies Used:

Technologies:

- Django
- Python
- Bootstrap
- HTML
- CSS

Tools:

- Git
- Visual Studio Code
- Pycharm

Software Requirement Specifications:

Users of the System:

- Three types of users should be able to use the system: **customer**, **employee** and **administrator**.
- Customers are users who visit the website and can create orders by customizing food, selecting products and entering customer details.
- Employees are the group of users that work with the ordering system on a daily basis. Employees will have their own accounts to log on to. They are the ones responsible for processing orders.
- The administrator, or super user, has the ultimate control of the system, he can add, change or delete ingredients and products, as well as add, change, or delete employee accounts.

Functional Requirements:

Customers:

1. Sign Up (only for new customer)
   Input: "SignUp" option selected.
   Output: customer prompted to enter the details.

2. Login
   Input: "Login" option selected.
   Output: customer prompted to enter the username and password.

3. Forgot password
   Input: "forgot password" option selected.
   Output: customer prompted to enter the email and new password.

4. Select food items
   State: The customer has logged in and the main menu has been displayed.
   Input: Items are selected customer feel free to order.
   Output: System will display selected items.

5. Changes to order
   Input: “go to cart” option selected.
   Output: customer can delete or add food item in order.

6. Review the order before submitting
   Input: “Order Place” option selected.
   State: Customer name, phone number, location (address) display or enter the all of information.
   Output: customer prompted to pay the bill.

7. Payment
   State: The different types of payment method are display.
   Input: choose any payment method.
   Output: customer prompted to enter the verification code if choose online payment.
   State: Display order no., payment details and confirmation of delivery.

8. Logout
   Input: “Logout” option selected.
   Output: you are successfully logout.
   State: System display login page.

Employees:

1. Login (Employee login page)
   Input: “Login” option selected.
   Output: Employee prompted to enter the username and password.

2. Modify Menu
   State: In the system all the items are displayed with their rates.
   Input: “Change” option selected.
   Output: Employee can make changings in menu like adding or removing food items which are not available and changings rate of items.

3. Order list
   Input: “Order list” option selected.
   State: System display all order details.
   Output: Employee can make changings like confirm order, prepared order, delivered order, not confirm order.

4. Logout
   Input: “Logout” option selected.
   Output: you are successfully logout.
   State: System display login page.

Administrators:

- Administrators must be able to use Employees all features.

1. Login (admin login page)
   Input: "Login" option selected.
   Output: admin prompted to enter the username and password.

2. Logout
   Input: "Logout" option selected.
   Output: you are successfully logout.
   State: System display login page.

Non-Functional Requirements:

1. **Portability**
   System running on one platform can easily be converted to run on another platform.

2. **Reliability**
   The ability of the system to behave consistently in a user-acceptable manner when operating within the environment for which the system was intended.

3. **Availability**
   The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs.

4. **Maintainability**
   A commercial database is used for maintaining the database and the application server takes care of the site.

5. **Security**
   Secure access of confidential data (customer information).

6. **User friendly**
   System should be easily used by the customer.

7. **Performance**
   Performance should be fast.

8. **Efficient**
   System should be efficient that it won't get hang if heavy traffic of order is placed.

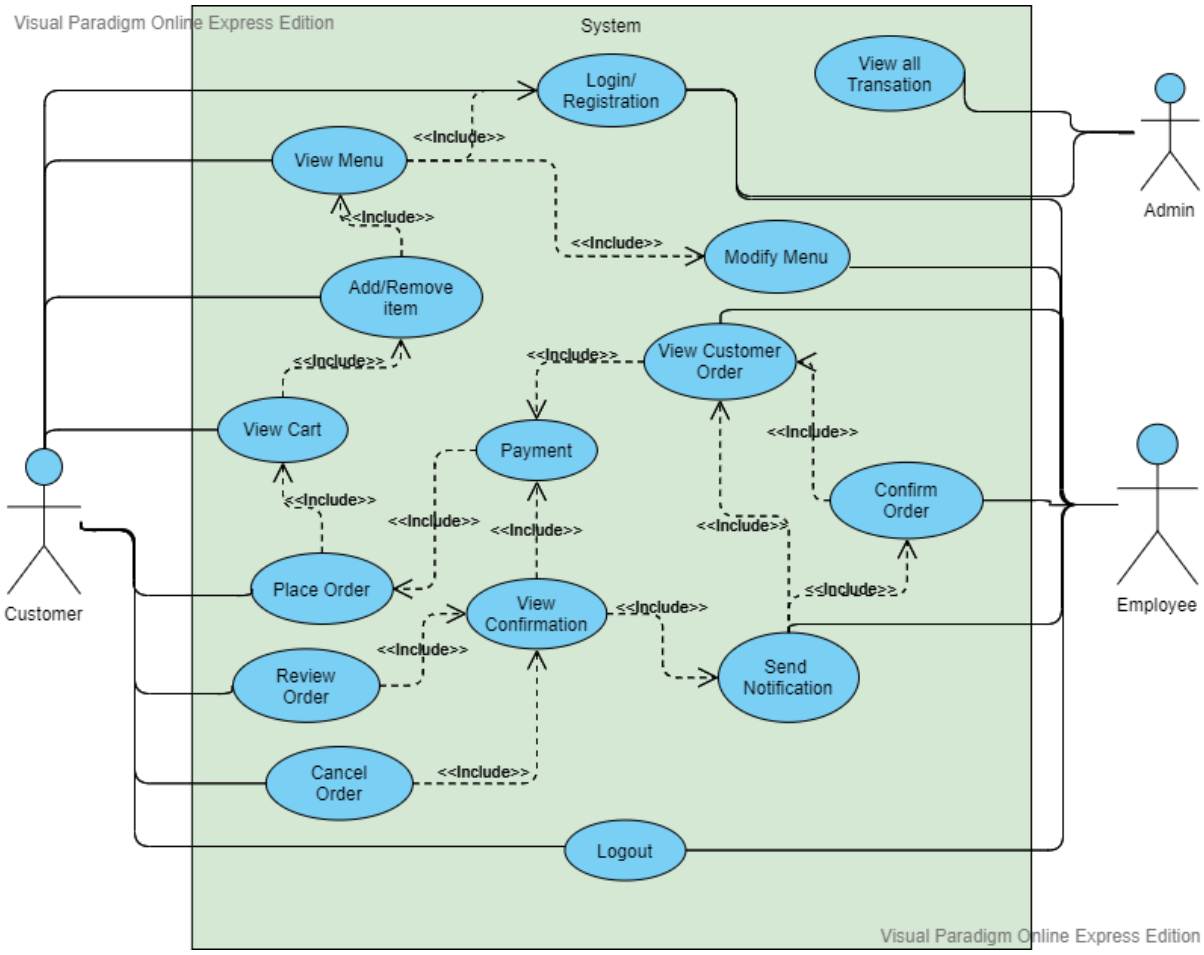
9. **Safety**
   Data in the database of system should not loss or damage.

10. **Privacy**
    Personal data of the system should not disclose to anyone.

Design Documents:

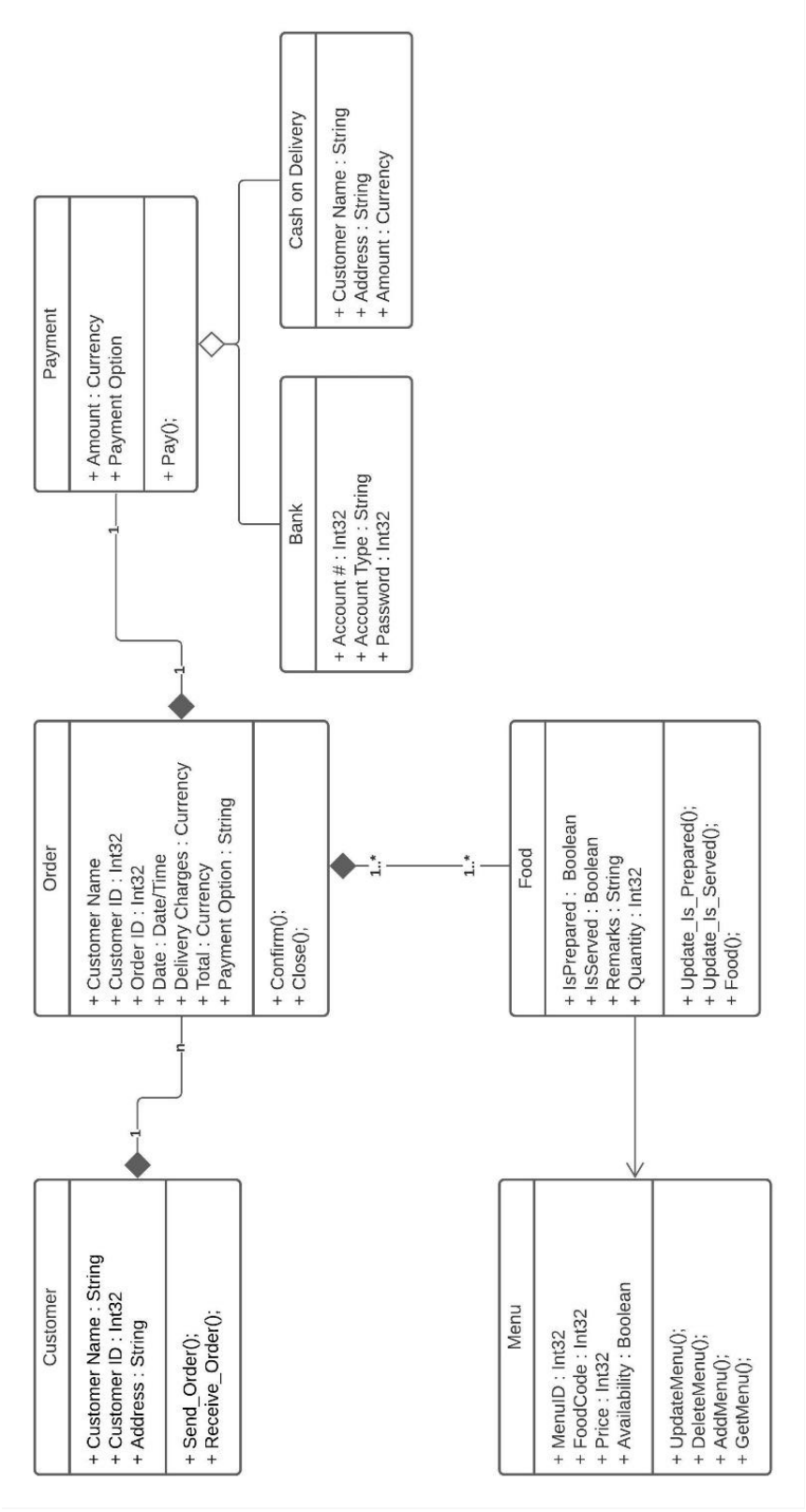
Use Case Diagram:

- Administrators(admin) can be able to use Employees all features.

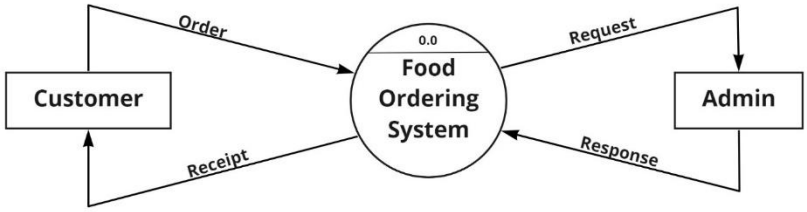


- In this use case diagram proposed have 3 actors, namely Customer, Employee and Admin.
- Customers can access the website to place orders and payments while Employee and Admin can access the website to receive orders, payments and making order report.
- Use case starts from login into the website with the customer's username and password, ordering food, making payment. Then Employee makes an order report based on customer order data.

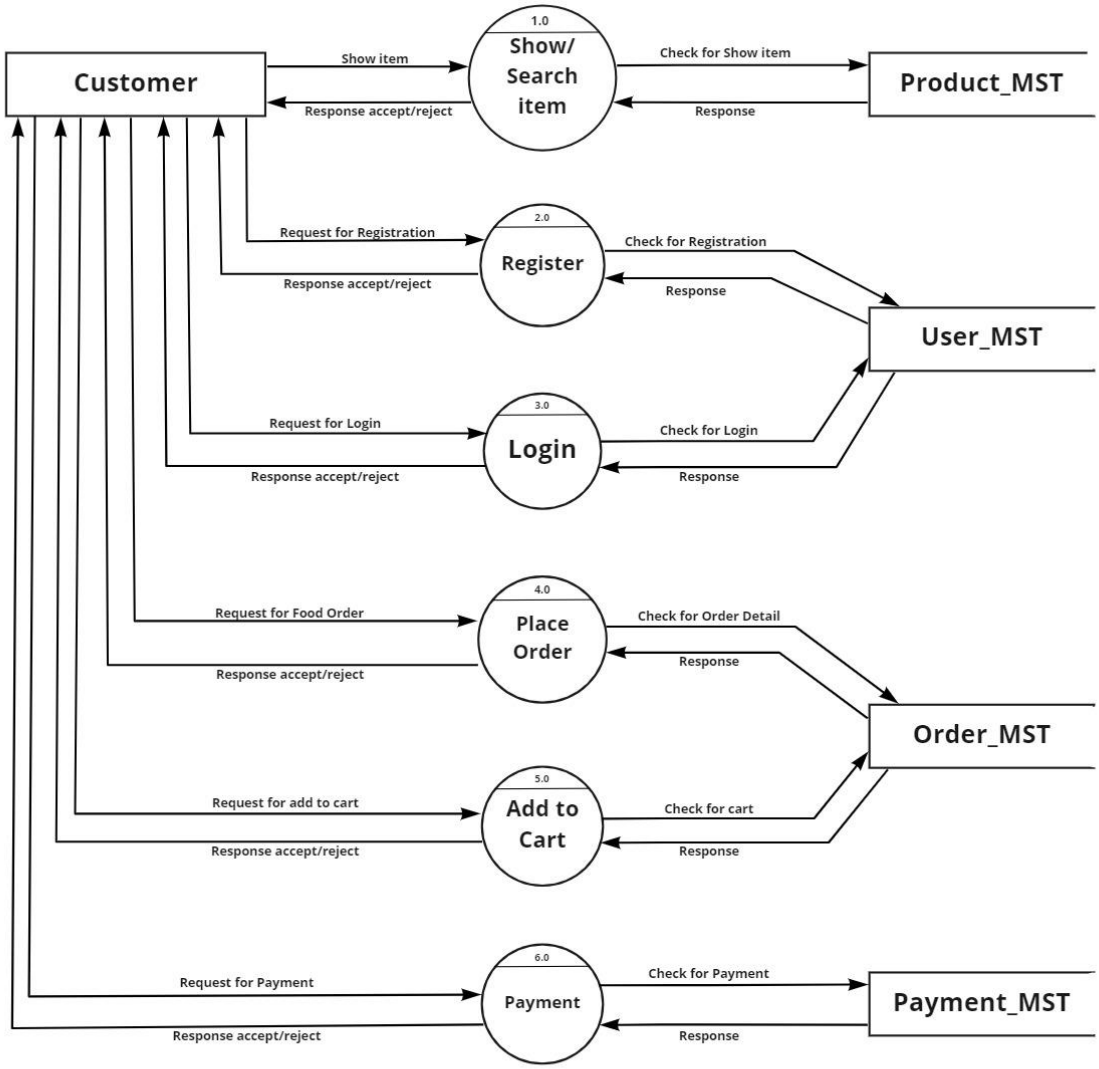
Class Diagram:



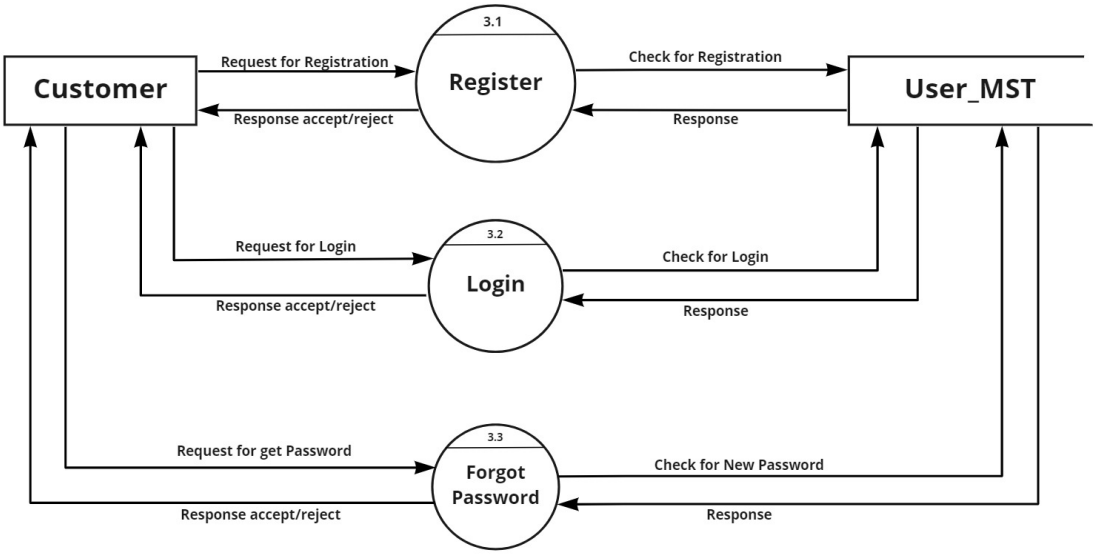
DFD Model:



Level 0 DFD

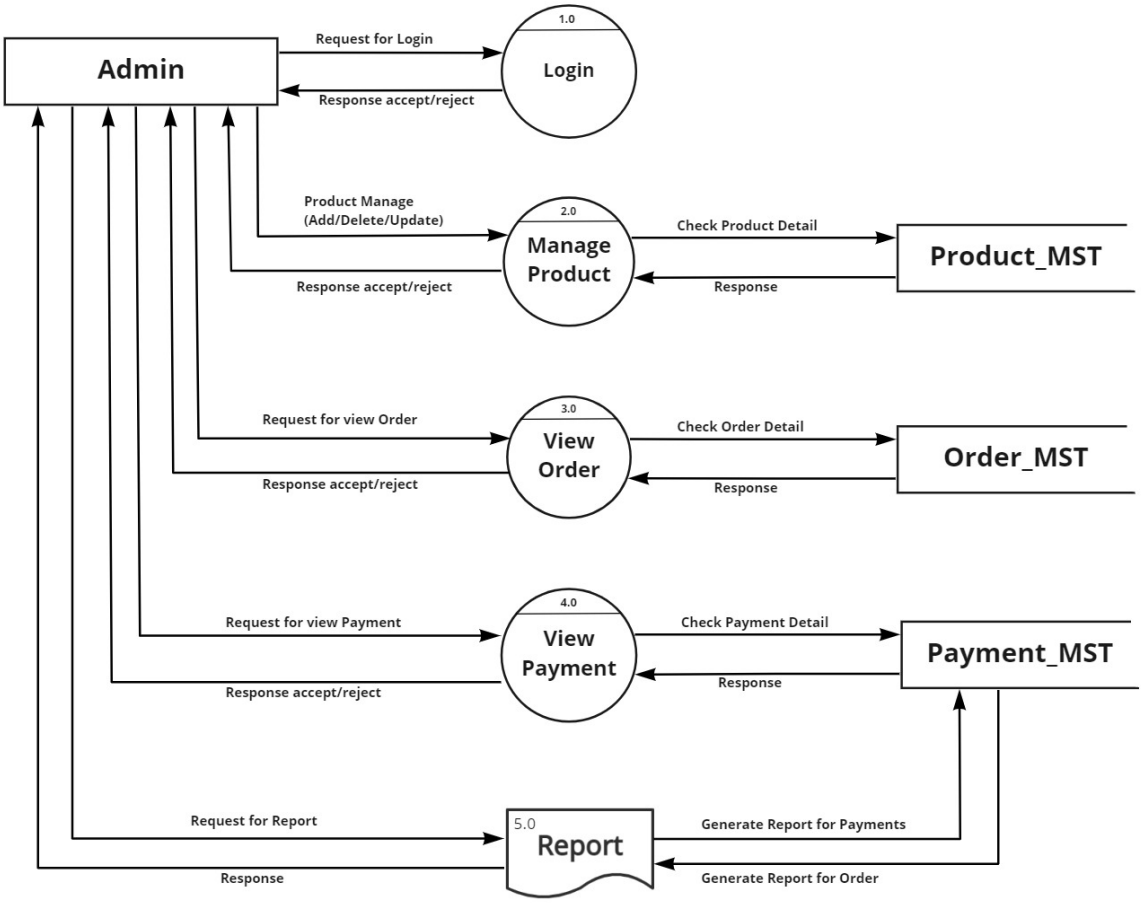


Level 1 DFD



Level 2 DFD(3.0)

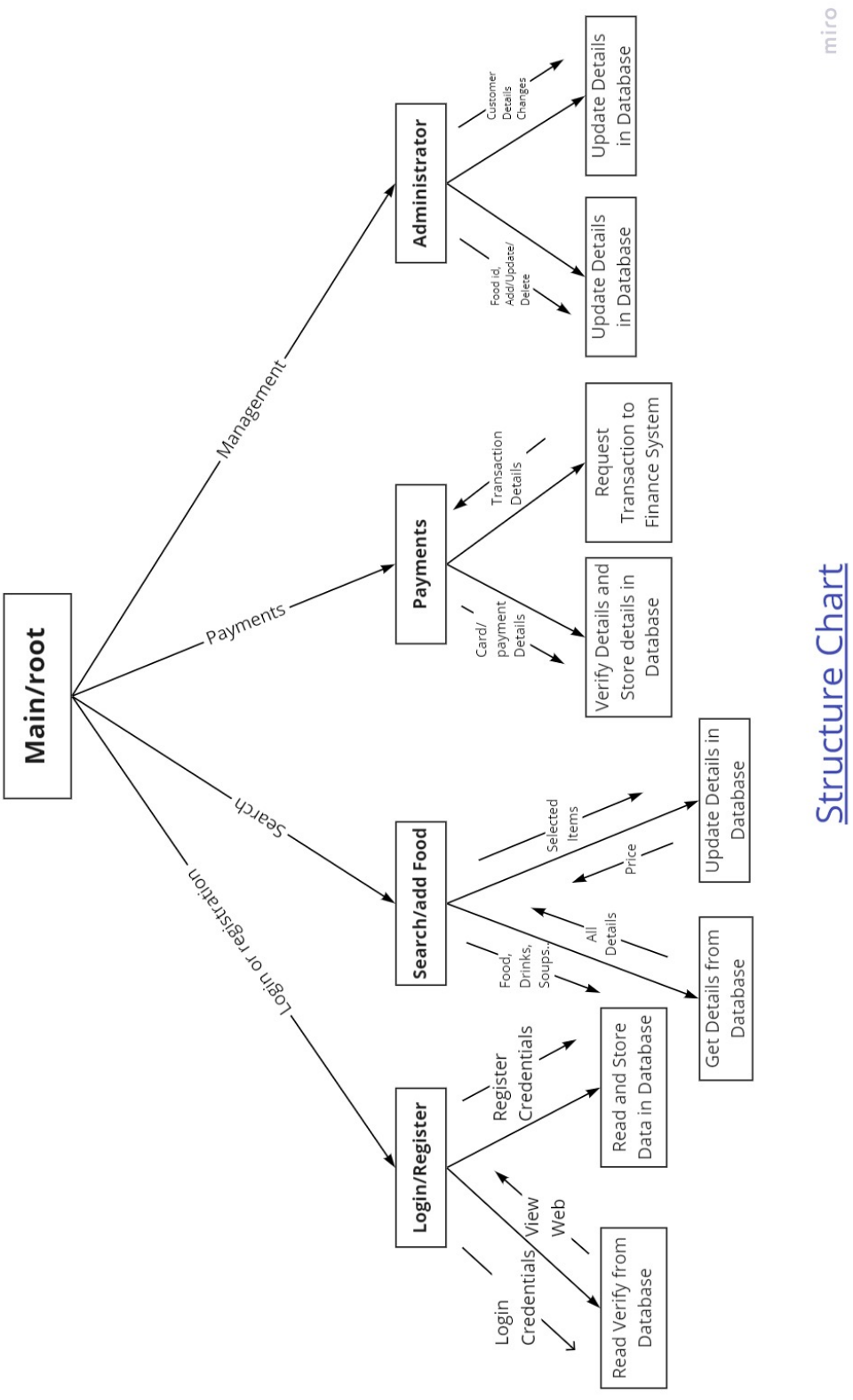
miro



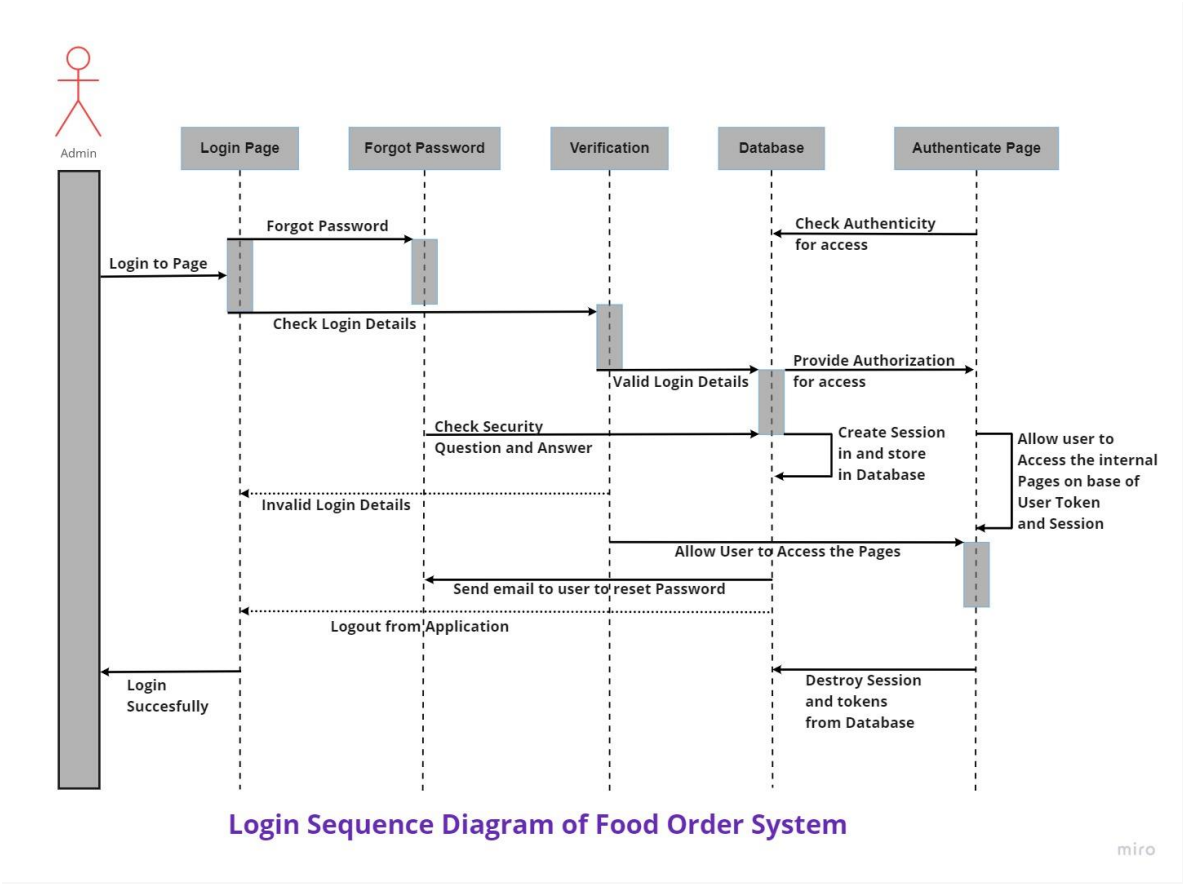
Admin side DFD

miro

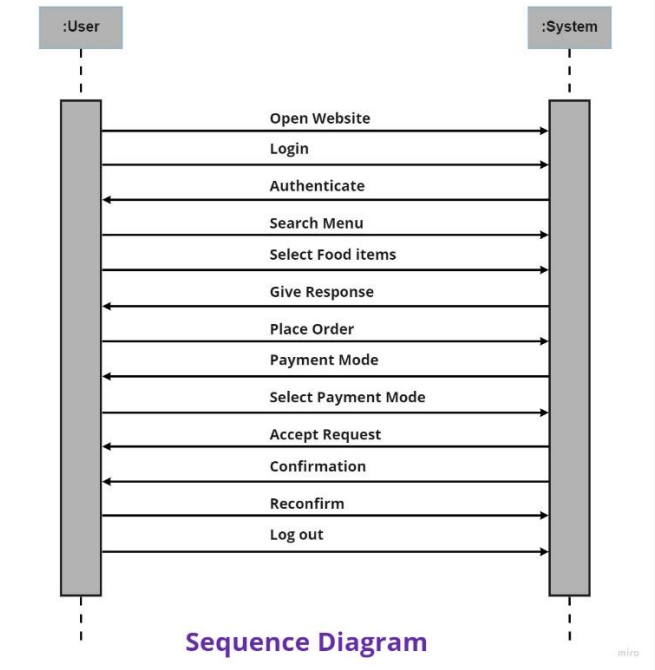
Structure Chart:



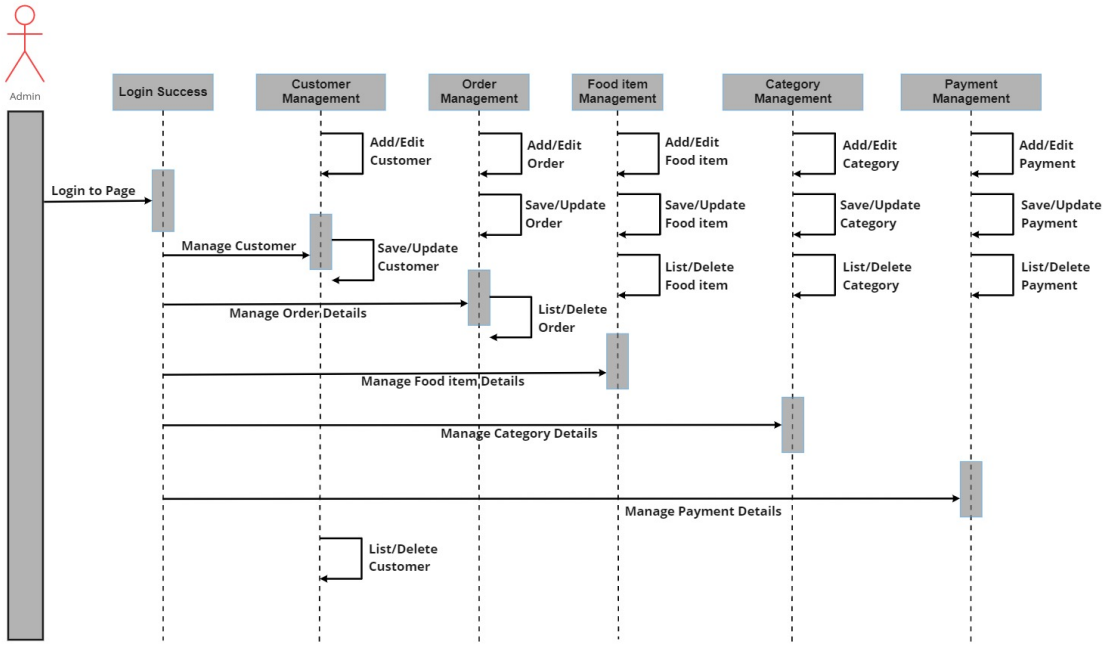
Sequence Diagrams:



Login Sequence Diagram of Food Order System



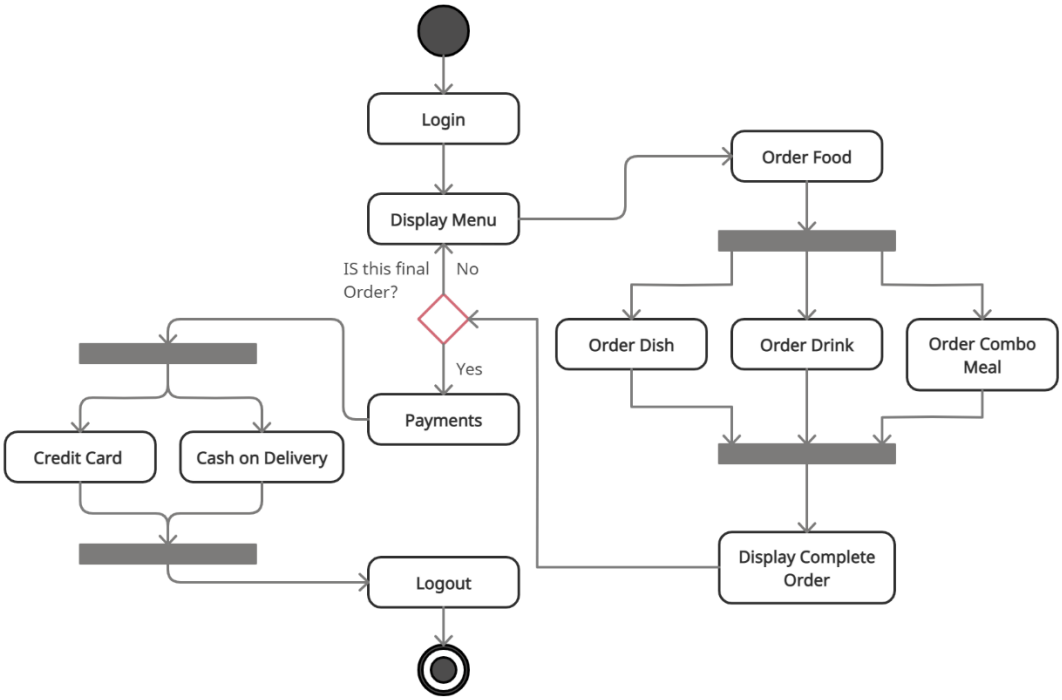
Sequence Diagram



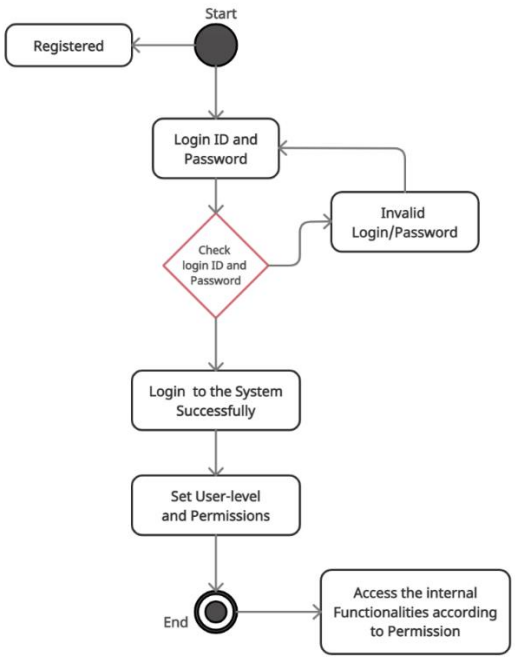
UML Sequence Diagram of Food Order System

miro

Activity Diagrams:



Activity Diagram



Login Activity Diagram

Implements Details:

Modules

- The system consists of 4 basic modules namely

    1. User Module
    2. Product Module
    3. Order Module
    4. Order Status Update Module

- Implementation is done using Django.

User Module

The main aim of the User Module is provide all the functionality related users. It track all the information of the customers. We have developed all type of operations of the customers. This is role based Module Where Admin can perform each and every operations on data but customer only view his/her data, so access level restrictions has also been implemented on the project.

Product Module

The main purpose for developing the Product Module is to manage Products category wise. All product will be managed by admin and Customer will be able to see product and buy them. Admin can see the list, change product details and also add or delete products.

Order Module

The main aim of the Order Module is receive all order details and display them. It is designed to be used only by restaurant employees (and admin), and provides the following functions: Retrieve new orders from the database and Display the orders in an easily readable, graphical way. Under “ViewOrder” a customer will be able to see only his/her order.

Order Status Update Module

The main aim of this Module is update all information related to order. Admin or employee can change or add order status. Customer only see his/her order status details. Under “Tracker” a customer will be able to see his/her order all status details.

Function prototypes

```python
def checkout(request):
    if request.method == "POST":
        items_json = request.POST.get('itemsJson', '')
        user_id = request.POST.get('user_id', '')
        name = request.POST.get('name', '')
        amount = request.POST.get('amount', '')
        email = request.POST.get('email', '')
        address = request.POST.get('address1', '') + " " + request.POST.get('address2', '')
        city = request.POST.get('city', '')
        state = request.POST.get('state', '')
        zip_code = request.POST.get('zip_code', '')
        phone = request.POST.get('phone', '')
        order = Orders(items_json=items_json, userId=user_id, name=name, email=email, address=address,
        city=city, state=state, zip_code=zip_code, phone=phone, amount=amount)
        order.save()
        update = OrderUpdate(order_id=order.order_id, update_desc="The Order has been Placed")
        update.save()
        thank = True
        id = order.order_id
        if 'onlinePay' in request.POST:
            # Request paytm to transfer the amount to your account after payment by user
            darshan_dict = {

                'MID': 'WorldP64425807474247',  # Your-Merchant-Id-Here
                'ORDER_ID': str(order.order_id),
                'TXN_AMOUNT': str(amount),
                'CUST_ID': email,
                'INDUSTRY_TYPE_ID': 'Retail',
                'WEBSITE': 'WEBSTAGING',
                'CHANNEL_ID': 'WEB',
                'CALLBACK_URL': 'http://127.0.0.1:8000/shop/handlerequest/',

            }
            darshan_dict['CHECKSUMHASH'] = Checksum.generate_checksum(darshan_dict, MERCHANT_KEY)
            return render(request, 'shop/paytm.html', {'darshan_dict': darshan_dict})
        elif 'cashOnDelivery' in request.POST:
            return render(request, 'shop/checkout.html', {'thank': thank, 'id': id})
    return render(request, 'shop/checkout.html')
```

Checkout

```python
def index(request):
    allProds = []
    catprods = Product.objects.values('category', 'id')
    cats = {item['category'] for item in catprods}
    for cat in cats:
        prod = Product.objects.filter(category=cat)
        n = len(prod)
        nSlides = n // 4 + ceil((n / 4) - (n // 4))
        allProds.append([prod, range(1, nSlides), nSlides])
    darshan = {'allProds': allProds}
    return render(request, 'shop/index.html', darshan)
```

View all Products

```python
def tracker(request):
    if request.method == "POST":
        orderId = request.POST.get('orderId', '')
        email = request.POST.get('email', '')
        name = request.POST.get('name', '')
        password = request.POST.get('password')
        user = authenticate(username=name, password=password)
        if user is not None:
            try:
                order = Orders.objects.filter(order_id=orderId, email=email)
                if len(order) > 0:
                    update = OrderUpdate.objects.filter(order_id=orderId)
                    updates = []
                    for item in update:
                        updates.append({'text': item.update_desc, 'time': item.timestamp})
                        response = json.dumps({"status": "success", "updates": updates, "itemsJson": order[0].items_json}, default=str)
                    return HttpResponse(response)
                else:
                    return HttpResponse('{"status":"noitem"}')
            except Exception as e:
                return HttpResponse('{"status":"error"}')
        else:
            return HttpResponse('{"status":"Invalid"}')
    return render(request, 'shop/tracker.html')
```

Track Order

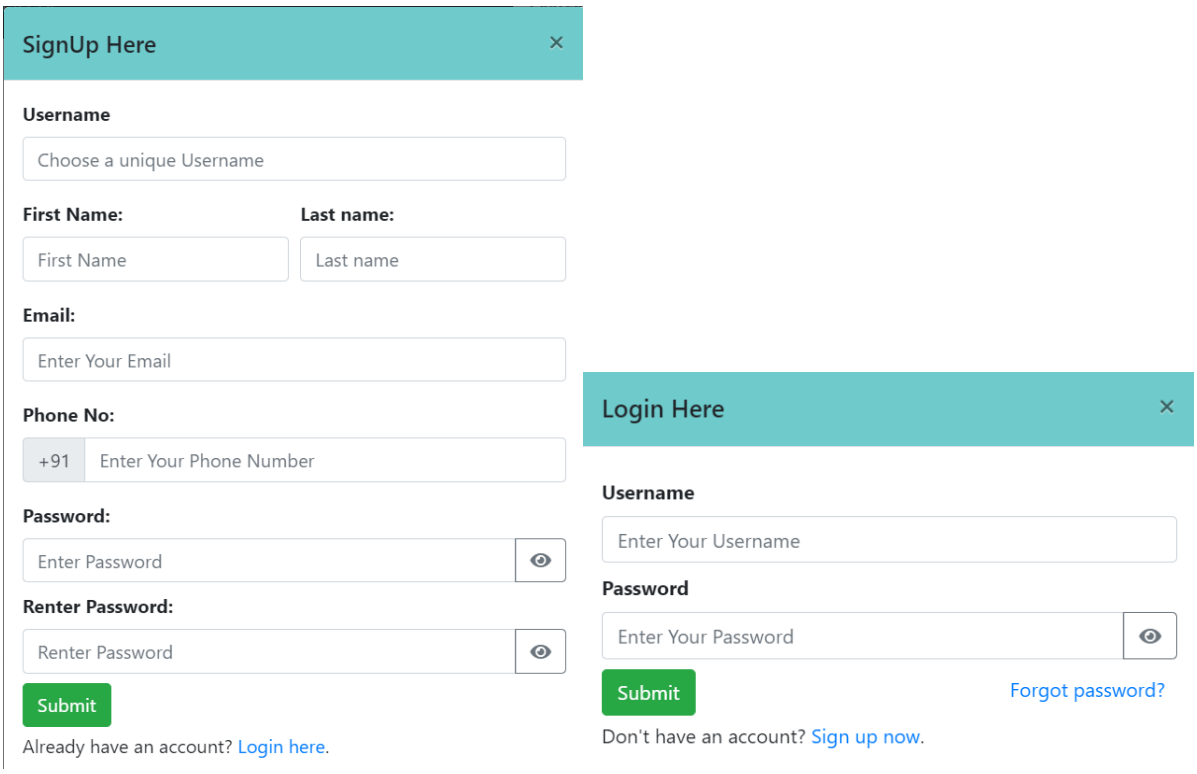
```python
def orderView(request):
    if request.user.is_authenticated:
        current_user = request.user
        orderHistory = Orders.objects.filter(userId=current_user.id)
        if len(orderHistory) == 0:
            messages.info(request, "You have not ordered.")
            return render(request, 'shop/orderView.html')
        return render(request, 'shop/orderView.html', {'orderHistory': orderHistory})
    return render(request, 'shop/orderView.html')
```

Order List

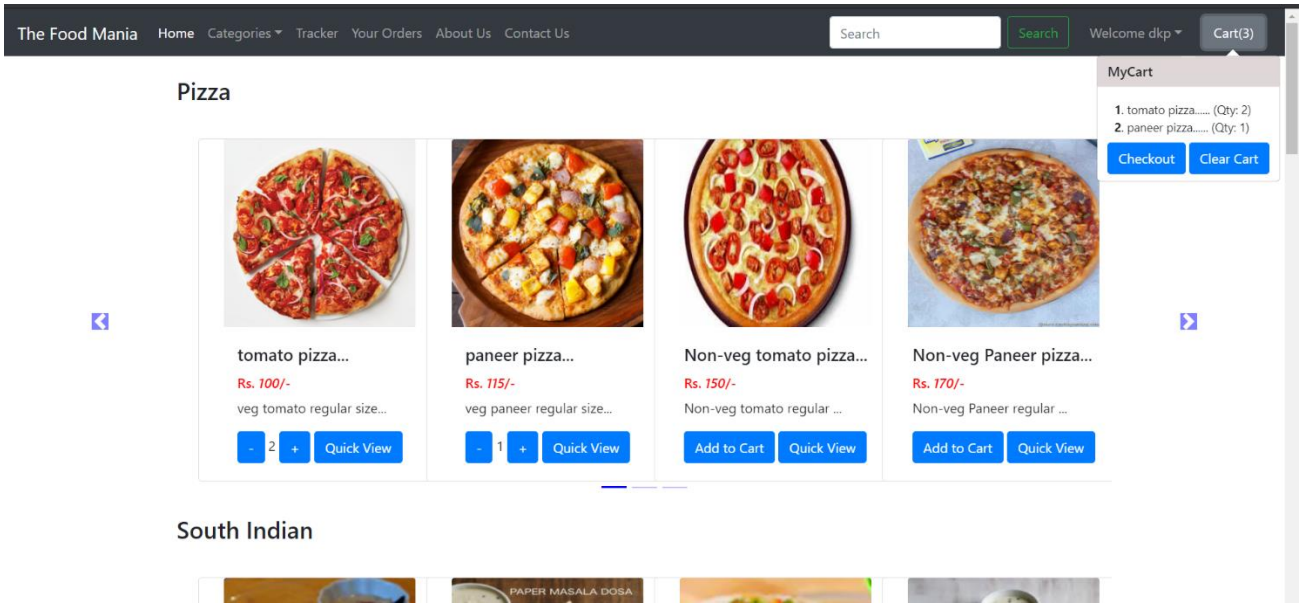
```python
def contact(request):
    thank = False
    if request.method == "POST":
        name = request.POST.get('name', '')
        email = request.POST.get('email', '')
        phone = request.POST.get('phone', '')
        desc = request.POST.get('desc', '')
        contact = Contact(name=name, email=email, phone=phone, desc=desc)
        contact.save()
        thank = True
        return render(request, 'shop/contact.html', {'thank': thank})
    return render(request, 'shop/contact.html', {'thank': thank})
```

Contact Admin

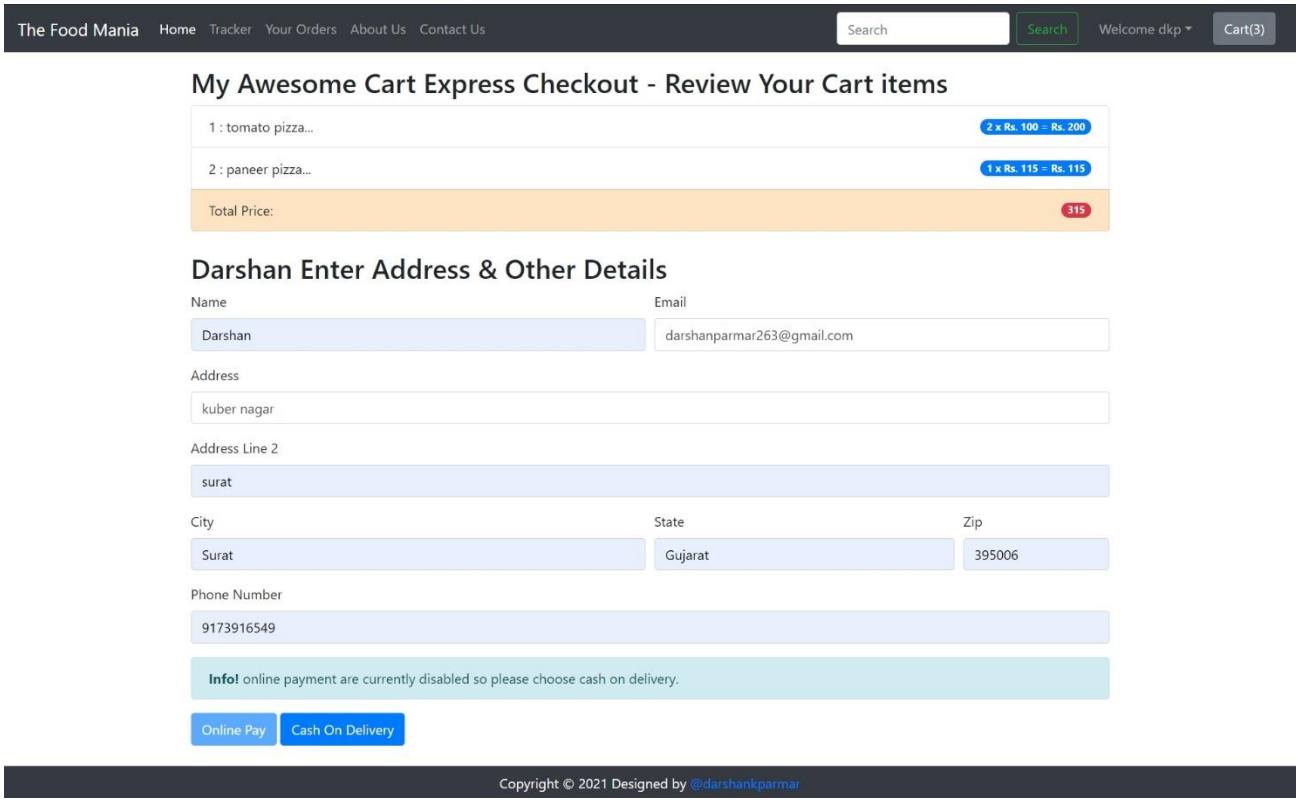
Work Flow/Layouts:



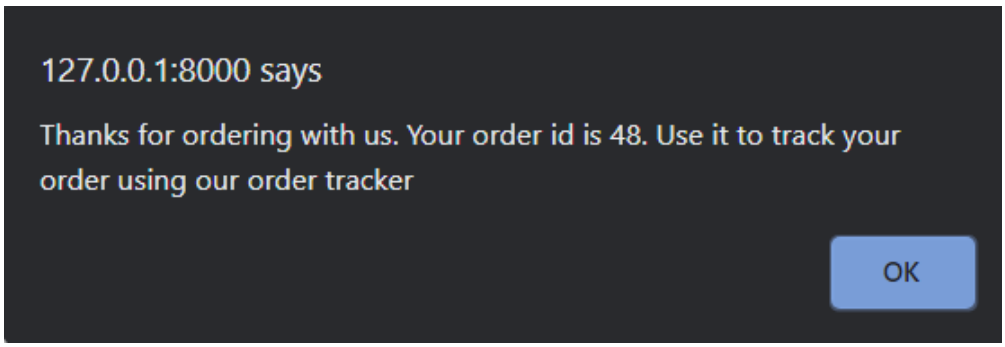
Login & SignUp module



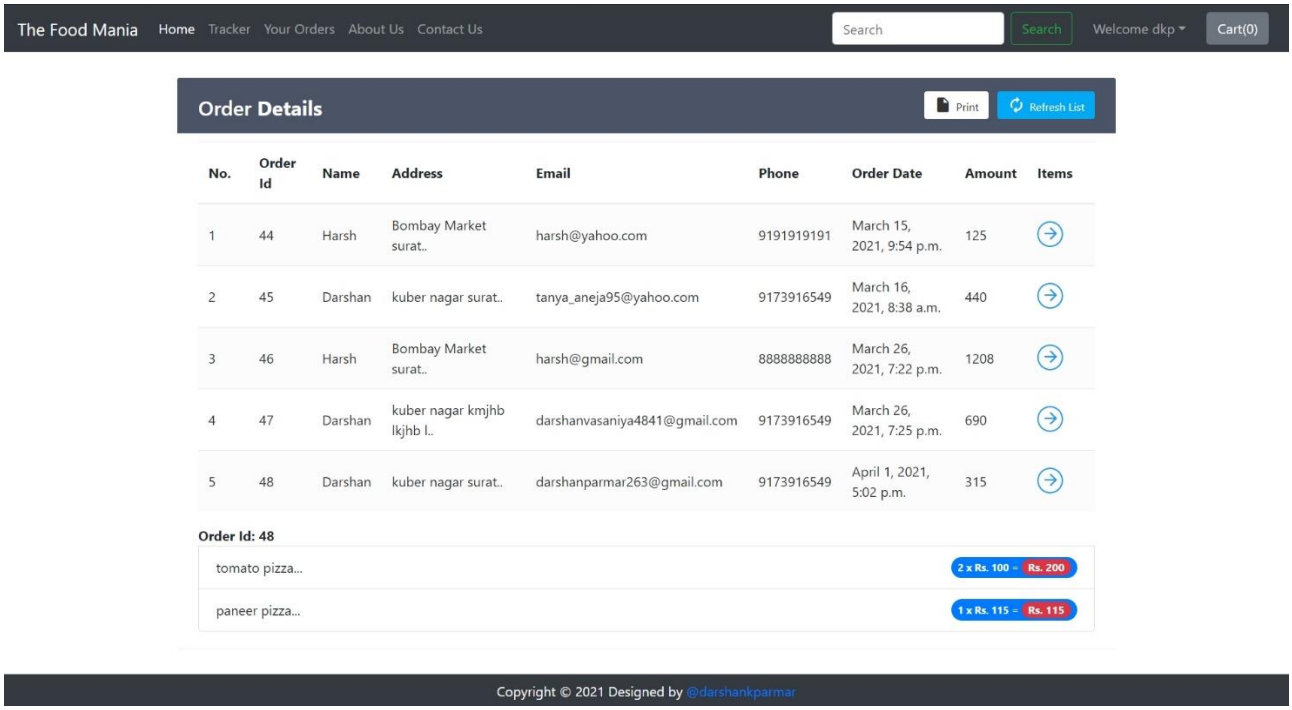
Home page



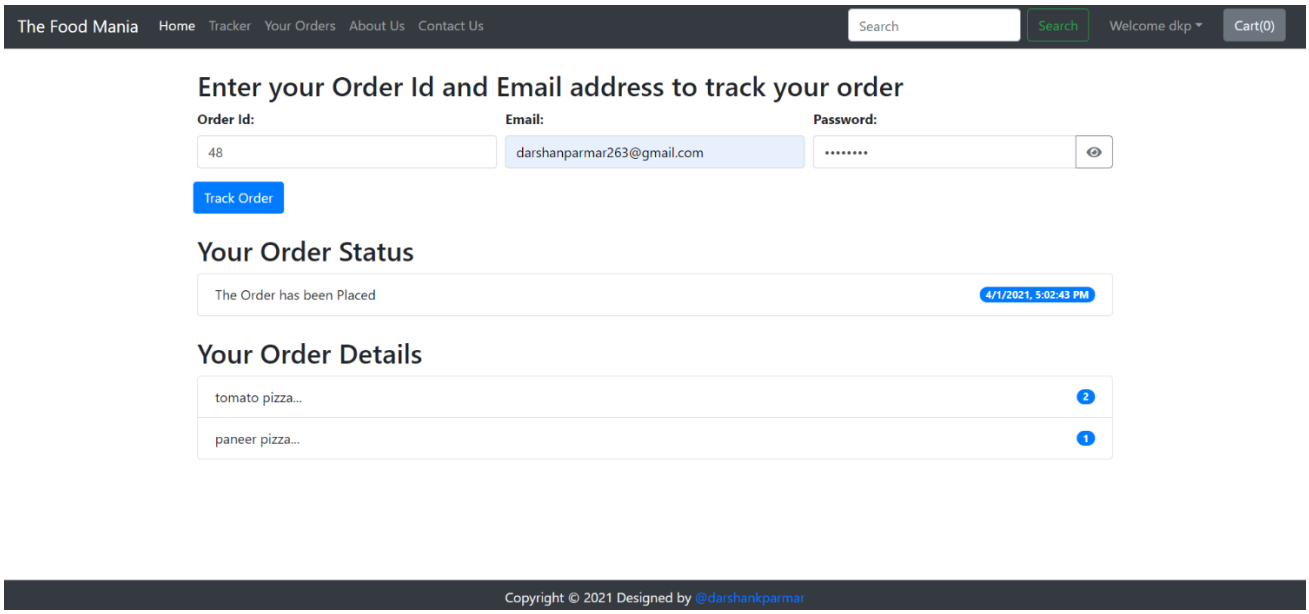
Chekout Page



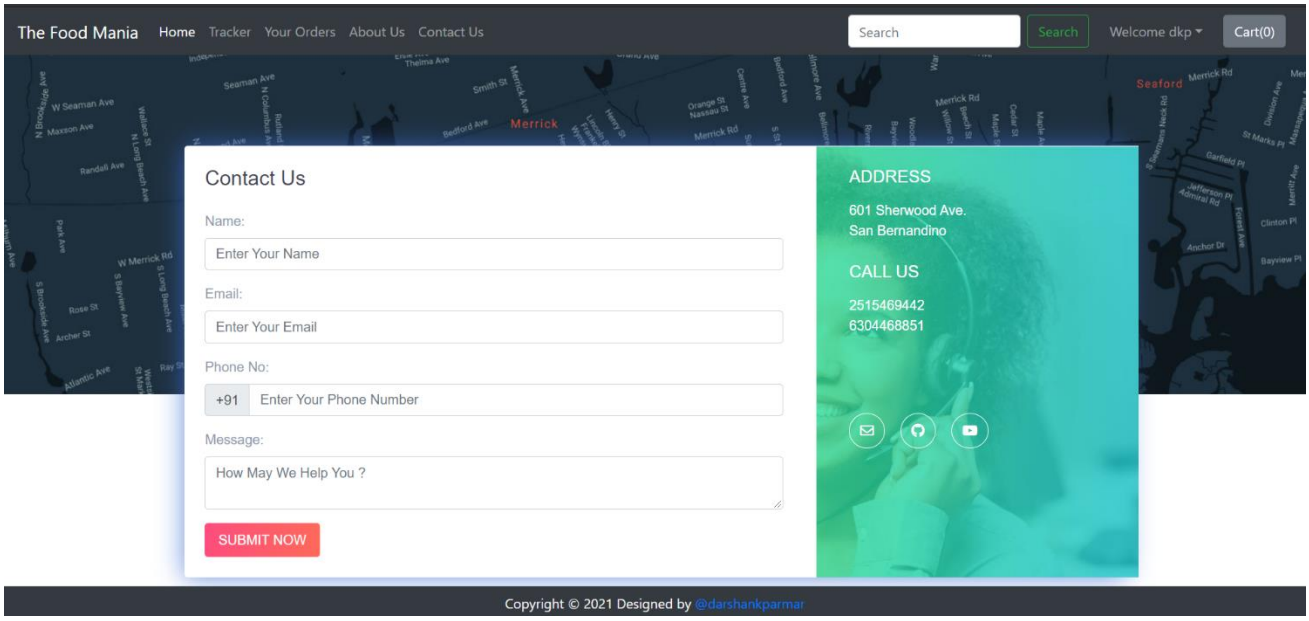
Confirm Order



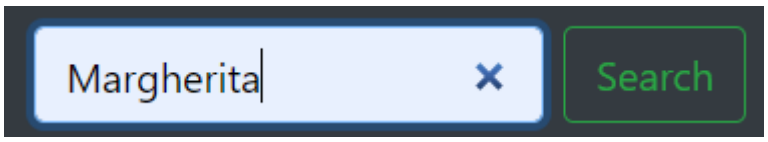
View Order Page



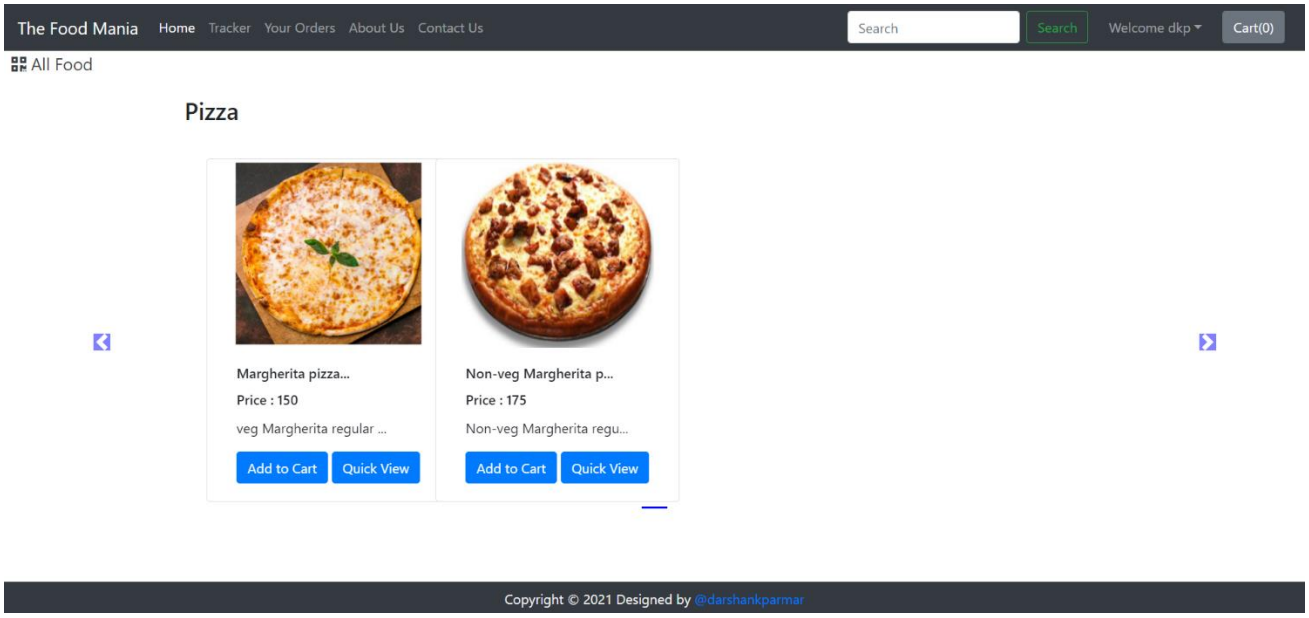
Tracker Page



Contact Us Page

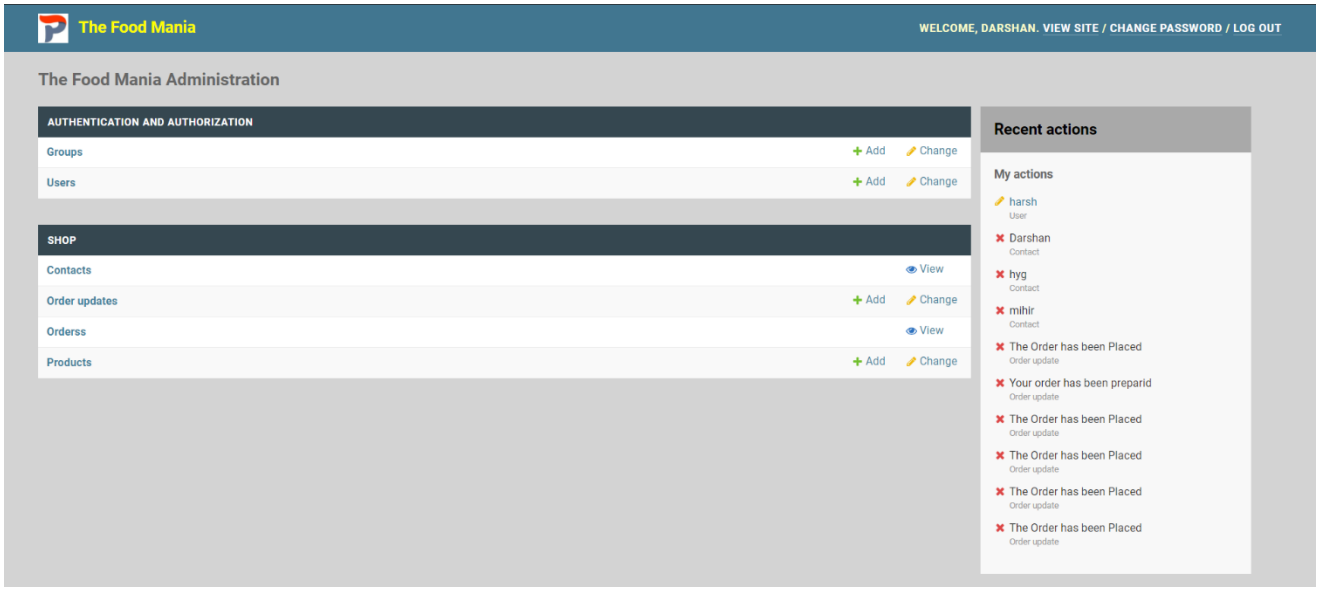


Search bar



Search Page

Admin Login Page



Admin Home Page

Conclusion:

After reviewing our work, the conclusion is that after many adjustments the system works. As good as it is now, there can still be made many adjustments/improvements. However in the time was given that two persons can work on this project, the overall results are satisfactory in our opinion. The report covers the entire course of the project and results are there were needed. The first weeks the work progressed slower than expected, then the pace was increased to finish on time.

For customers, web-based ordering system can make it easier to order food without having to visit the restaurants so that customers can save time and costs. For admin, they can serve customers optimally in ordering their food and making the order report easier. Payment methods can also be done by customers through a system that is available on the web to facilitate customers in paying for their orders.

Limitations and Future Extensions:

The following section describes the work that will be implemented with future releases of the software.

- Customize orders: Allow customers to customize food orders
- Enhance User Interface by adding more user interactive features. Provide Deals and promotional Offer details to home page. Provide Recipes of the Week/Day to Home Page.
- Payment Options: Add different payment options such as PayPal, Cash, Gift Cards etc. Allow to save payment details for future use.
- Allow to process an order as a Guest
- Order Process Estimate: Provide customer a visual graphical order status
- Restaurant Locator: Allow to find and choose a nearby restaurant

Reference / Bibliography:

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

- https://getbootstrap.com/
- https://www.djangoproject.com/
- https://github.com/
- http://stackoverflow.com/
- https://codewithharry.com/