



Final Report

[Online Food Ordering System]

ICT 2212 Skills Development Project II
Bachelor of Information and Communication Technology
(BICT) Degree Programme

Department of Information and Communication Technology
Faculty of Technology
Rajarata University of Sri Lanka
Mihintale

Details of the Project

Project Title : Online Food Ordering System(OFOS)

Group Number : 07

Group Name : OMICRON

Submission Date : 2023.06.12

Group Members :

Student Name	Index Number	Signature
1. M.U.K.Gunawardhana	ITT/2017/2018/030	0718
2. K.B.Shalika	ITT/2017/2018/069	0754
3.M.G.Y.D.Gamage	ITT/2017/2018/028	0716
4. W.M.D.D.B.Wijesooriya	ITT/2017/2018/088	0769
5. S.Tharsan	ITT/2017/2018/075	0758

Supervisor(s) :

Name : Mrs. Dinusha Premasiri

Designation : Lecturer (Probationary)

Department : Department of Information Communication and Technology

Email : dmpremas@tech.rjt.ac.lk

Signature : _____ Date: _____

Content

1. Chapter 1 – Introduction.....	1-4
1.1. Introduction.....	1
1.2. Problem Statement.....	1
1.3. Aim.....	1
1.4. Objectives.....	2
1.5. Significance of the project.....	2
1.6. Project Work Plan.....	3
1.7. Assumptions and Limitations.....	4
2. Chapter 2 – Approach.....	5-8
2.1. Scope.....	5-7
2.1.1. List of specific project goals.....	5
2.1.2. List of deliverables.....	5
2.1.3. List of features.....	5
2.1.4. List of functions.....	5
2.1.5. List of tasks.....	6
2.1.6. List of deadlines.....	7
2.1.7. List of ultimately costs.	7
2.2. Organization.....	7
2.3. Project type.....	7
2.4. End users of the project.....	7
2.5. Feasibility study.....	7-8
2.5.1. Technical feasibility.....	8
2.5.2. Operational feasibility.....	8
2.5.3. Organizational feasibility.....	8
2.6. Rich picture.....	8
3. Chapter 3 – Analysis.....	9-13
3.1. Data Gathering.....	9
3.2. Analysis of the gathered data.....	10
3.3. Extant system.....	11
3.3.1. Process of the current system.....	11
3.3.2. Problems and limitations of the extant system.....	11
3.4. Requirement Specification.....	12-13
3.4.1. Functional requirements.....	12
3.4.2. Nonfunctional requirements.....	13
4. Chapter 4: Design.....	14-42
4.1. System Model.....	14
4.1.1. Use case diagram.....	14
4.1.2. Class diagram.....	15
4.1.3. Activity diagrams.....	15
4.1.4. Sequence diagrams.....	16
4.2. System Architecture.....	17

4.2.1. Presentation Layer.....	17
4.2.2. Application Layer.....	17
4.2.3. Database Layer.....	17
4.3. Database Design.....	18
4.3.1. Entity relationship diagram.....	18
4.3.2. Schema diagram.....	
4.4. Interface Design.....	19
4.4.1. Considerations when designing interfaces.....	19
4.4.2. Used tools.....	19
4.4.3. Screen shots of the interfaces.....	20-42
5. Chapter 5: Development.....	43
5.1. Solution.....	43
5.1.1. Hardware.....	43
5.1.2. Software.....	43
5.2. Technology adopted.....	43
5.2.1. Hardware equipments.....	43
5.2.2. Software tools.....	43
5.2.3. Frameworks.....	43
5.2.4. Integrated Development Environment (IDE)	43
6. Chapter 6: Testing and evaluation.....	44-46
6.1. Testing plan.....	44
6.2. Testing.....	45
6.2.1. Unit testing.....	45
6.2.2. Integrated testing.....	45
6.2.3. System testing.....	45
6.2.4. Acceptance testing.....	45
6.3. Test results and conclusion of testing.....	46
7. Chapter 7: Conclusion.....	47-49
7.1. Conclusions of the project.....	47
7.2. Lessons learned and skills earned.....	48
7.3. Recommendations for improvements.....	49
8. Chapter 8: References and Appendixes.....	50-51
8.1. Appendixes	50
8.2. References.....	51

List Of Figures

1. Figure 01-6. Project Work Plan.....	3
2. Figure 02-2.6 Rich picture.....	8
3. Figure 03- 4.1.1 Use case diagram.....	14
4. Figure 04- 4.1.2 Class diagram.....	15
5. Figure 05- 4.1.3 Activity diagram.....	15
6. Figure 06- 4.1.4 Sequence diagram for user	16
7. Figure 07-4.1.4 Sequence diagram for placing order.....	16
8. Figure 08-4.1.4 Sequence diagram for delivery	17
9. Figure 09- 4.2. System Architecture.....	17
10. Figure 10-4.2 Web request responsive cycle.....	18
11. Figure 11-4.2 Architecture Diagram with layers.....	18
12. Figure 12- 4.3.1. Entity relationship diagram.....	19
13. Figure 13- 4.4.3 Login page.....	20
14. Figure 14- 4.4.3 Registration page.....	20
15. Figure 15-4.4.3 Profile of distributor.....	23
16. Figure 16- 4.4.3 Order details page.....	24
17. Figure 17-4.4.3 orders.....	25
18. Figure 18-4.4.3 Home page of web application-1.....	27
19. Figure 19-4.4.3 Home page of web application-2.....	28
20. Figure 20-4.4.3 Home page of web application-3.....	28
21. Figure 21-4.4.3 Dish page of web application.....	31
22. Figure 22-4.4.3 Dish page of web application.....	31
23. Figure 23-4.4.3 Dish page of web application.....	32
24. Figure 24-4.4.3 About us page of web application.....	33
25. Figure 25-4.4.3 Contact Us page of web application.....	34
26. Figure 26-4.4.3 Login page of web application.....	35
27. Figure 27-4.4.3 Signup page of web application.....	36
28. Figure 28-4.4.3 Add to cart page of web application.....	37
29. Figure 29-4.4.3 Cart page of web application-1.....	38
30. Figure 30-4.4.3 Cart page of web application-2.....	39

Chapter 1 – Introduction

1. Introduction

In the past, the food requirement was completed by people who live in the Sri Lanka using several ways but today people in competitive world find shortest path for fulfilling the food requirement with new technology. In that situation online food ordering concept is invented. This concept is very easy and save money and time of people. The online Food ordering system that we are proposing here mostly simplifies the ordering process for both the customer and the restaurant or the hotels. This System will be an interactive and up-to-date menu with all available options for all food items.

There is web application and mobile application .The web application interact with user or customer and mobile application interacts with distributors(Restaurant Employees) who are delivering foods. Customer(User) can choose one or more food items to place an order which will land in the Cart. And that Cart will contain all the ordered items for delivering the order to the customer. And Customers can view all the order details in the cart before checking out. In the end, the customer gets order confirmation details.

Once the order will be placed it will enter the database system and retrieved that information. This System will allow Restaurant Employees(Distributors) to rapidly go through the orders as they are received and process all orders efficiently for delivering and effectively with minimal delays and confusion to make happy the customers.

2. Problem Statement

Living in a busy lifestyle, we looking for solutions using new technology to make our daily work easier. Accordingly, one of the problems that arose is that it is difficult to prepare food in busy times and when we need to satisfy our dieries,and we want to satisfy his desires while working office or staying outside, we have to bring orders according their needs to a market to fulfill his needs.

3. Aim

The aim of an online food ordering system is to provide a convenient and user-friendly platform for customers to order food from various restaurants or food establishments.

4. Objectives

Online food ordering systems aim to provide a convenient and seamless experience for customers when ordering food online. This includes easy navigation, a user-friendly interface, and multiple ordering options. Additionally, businesses aim to reduce errors and improve accuracy of order processing, leading to higher customer satisfaction.

Improve Operational Efficiency: An online food ordering system should optimize operational processes for the restaurant or food establishment. This includes automating tasks such as order management, inventory tracking, and delivery coordination, leading to improved efficiency and reduced costs.

Expand Customer Reach: By going online, businesses aim to expand their customer base beyond the traditional brick-and-mortar setup. The objective is to reach a wider audience and attract new customers who prefer the convenience of online ordering.

Enhance Brand Image and Customer Loyalty: A well-designed online food ordering system can contribute to enhancing the brand image and building customer loyalty. By providing a positive and seamless experience, customers are more likely to trust and choose the brand repeatedly.

5. Significance of the project

The customer's personal detail is collected only when he places an order. Otherwise, the customer will not receive his details when registering to the system. The information obtained in this way is stored very securely in the memory.

6. Project Work Plan

Gantt Chart

Task	October				November				December				January				February				March			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Planning																								
Requirement Analysis																								
Design																								
Testing																								
Implementation																								
Maintenance																								

Figure 01-6. Project Work Plan

7. Assumptions and Limitations

Internet accessibility: The system makes the assumption that consumers can connect to the online ordering system or application and have access to the internet.

User Devices: In order to access the online meal ordering system, users are presumed to have suitable devices such as smartphones, tablets, or PCs with internet connection.

System Accessibility: The system takes the position that the online application or platform is usable by a variety of users, including those with impairments. To provide a welcoming user experience, it must adhere to accessibility requirements.

Menu and meal Availability: The system deems the online menu to be an accurate representation of the meal options, costs, and any other pertinent information. Additionally, it is supposing that the mentioned products are in stock or are accessible for ordering.

Technical Problems: Systems for ordering meals online depend on reliable internet connections and working technology. Any technical problems might interfere with the system's availability and prohibit customers from making purchases, such as server crashes, network outages, or device malfunctions.

Limited Menu Visibility: Compared to actual menus in restaurants, online menus could not offer as much information or aesthetic appeal. Users could find it challenging to judge culinary products' quality and presentation only based on textual descriptions or still photographs.

Orders might be filled inaccurately for a variety of reasons, including the wrong foods, missing goods, or the wrong quantity. Despite attempts to reduce mistakes, misunderstandings or human error during order preparation or delivery might result in unhappy customers.

2. Chapter 2 – Approach

2.1.Scope

2.1.1. List of specific project goals

- provide user-friendly interfaces for users to easily browse.
- Implement a secure information of user protection
- Provide customers(users) with real time order tracking functionality
- Create an effective and efficient order management system
- Develop a mobile application that is compatible with Android platforms
- Provide best delivery service for users

2.1.2. List of deliverables

- User Registration and Authentication
- Menu and Restaurant Information
- Order Placement and Management
- Real-time Order Tracking
- Delivery Partner Integration
- Feedback and Rating System
- Mobile Application
- Administrative Dashboard

2.1.3. List of features

- User Registration and Profiles:-Allows to users to register and create profiles.
- Menu selection:Here provide menu items and prices in home page and dish page
- Customization Options: Enable users to customize their orders by selecting ingredients
- Order Placement and Checkout:users can add items to cart and review their orders and proceed to checkout
- Delivery management: Streamline the delivery process by assigning orders to delivery personnel, optimizing routes.
- Reviews and Ratings: Allow users to provide feedback and ratings for restaurant
- Admin Dashboard: Provide a comprehensive administrative dashboard for restaurant owners and platform administrators to manage menus.

2.1.4. List of functions

- User Registration and Authentication
- Menu Management
- Order Placement
- Order Confirmation and Notification
- Order Processing and Management

- Delivery Management of mobile application
- System Administration

2.1.5. List of deadlines

- Project Planning and Requirements Gathering:-
 - Identify goal and objectives and scope of project and outcomes
 - We allocate resources and timelines for it
- System Design and Architecture
 - We arranged meeting physically and design overall system architecture and database. There are five members in the group and divide the parts for members.
 - We have created github repository and assign works for members.
 - We created prototypes and dataflow diagrams to visualize the system
 - Define the technology stack and frameworks to be used
- Front-end and Back-end Development:
 - Develop the front-end user interfaces, including menu browsing, order placement, and tracking functionalities
 - Implement the back-end systems, such as order management, Storing information and integration with third-party services.
 - Perform thorough testing and debugging throughout the development process.
- Integration and Testing:
 - Integrate different system components and ensure smooth communication and data flow between them.
 - Conduct comprehensive testing, including functional testing, usability testing, security testing, and performance testing.
 - Identify and resolve any issues or bugs that arise during testing.
- Deployment
 - Set up the necessary infrastructure, servers, and databases for hosting the system.
 - Coordinate with restaurant partners, delivery services, and other stakeholders for a smooth launch

2.1.6. List of ultimately costs.

There is no any costs because development team is not hired .we complete all the works by us.we design and create all pages and pages are not bought as well as mobile application is created by boys who joined to group.

2.2 Organization

This system will be implement in restaurants which are situated in sri lanaka.or This system suited for places where storing and delivering foods

2.3 Project type

We created Web application and mobile application .web application can get user requirements regarding foods requirements and mobile application used to deliver foods to customers.That's why we decided to this project.

2.4 End users of the project

- Customers: Customers are the primary end users of the system.
- Restaurant Owners/Managers: Restaurant owners or managers are admins of this system
- Distributors :They delivery the food for relevant locations.

2.5 Feasibility study

2.5.1 Technical feasibility

We can consider technical feasibility under several key points regarding online food ordering system.technical feasibility depend on the technology that we had used for system building.

This system is used with people who have interact with new technology and it is not suitable for rural areas. The system may need to integrate with various third-party services, such as payment gateways, SMS notification providers, mapping services for delivery tracking, or restaurant POS systems. Ensuring compatibility and smooth integration with these services is crucial for seamless operation

2.5.2 Operational feasibility

The operational feasibility depend on the resources that we have. target users (customers and delivery personnel) are willing to adopt and use the online food ordering system.As well as this system affect for the market and improve business strategies . Evaluating the impact of the system on existing operational processes. Identify any modifications or adjustments required in the restaurant's workflow, staff roles, and responsibilities to accommodate the online ordering system effectively.

2.5.3 Organizational feasibility

Assess the readiness and willingness of the organization to embrace and implement the online food ordering and delivery system. Evaluate the organization's culture, flexibility, and openness to change to ensure a smooth adoption of the new technology. Evaluate the availability of resources, both financial and human, to support the development, implementation, and ongoing operation of the system. Determine if there is sufficient support and commitment from management to drive the project forward. Obtain buy-in from key stakeholders and decision-makers to ensure the necessary resources, funding, and organizational alignment.

2.6 Rich picture

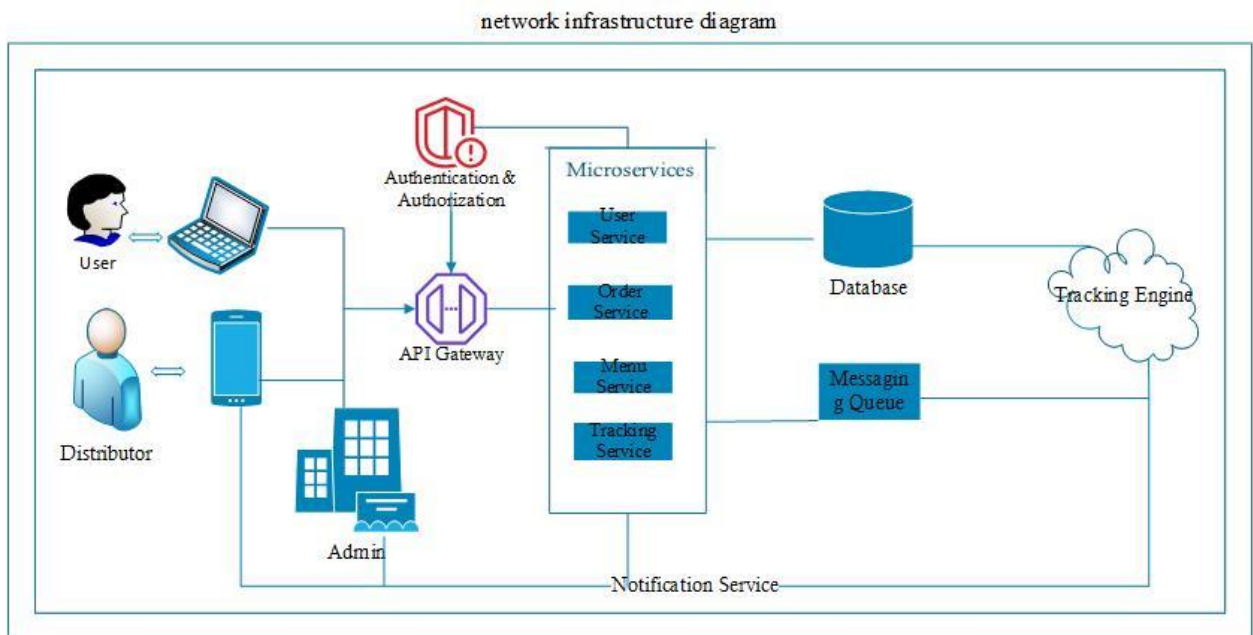


Figure 02-2.6 Rich picture

3.Chapter 3 – Analysis

3.1 Data Gathering

Fact gathering techniques used

- **Market Research:** Conducted market research to understand the current trends, competition, and customer preferences in the online food ordering industry. This provides insights into the features, functionalities, and user experience expectations that should be considered in the system design.
- **Document Analysis:** Reviewed relevant documents, such as existing food ordering systems, menus, invoices, and customer feedback. Analyzing these documents provides valuable information about current practices, challenges, and areas for improvement.
- **Interviews:** Conducted interviews with stakeholders such as customers, restaurant owners, delivery personnel, and system administrators. This allows to directly interact with individuals and gather their perspectives, requirements, and feedback.
- **Observation:** Observe the existing food ordering processes at restaurants or hotels to understand their operations, workflows, and pain points. This helps identify areas where the system can streamline or enhance efficiency.
- **Focus Groups:** Organized focus groups consisting of potential users or customers to facilitate group discussions and gather insights. This technique encourages participants to share their experiences, expectations, and suggestions, leading to a more comprehensive understanding of user needs.

Obstacles faces by data gathering process

- **Limited Stakeholder Availability:** Stakeholders, such as customers, restaurant owners, or administrators, have busy schedules or limited availability, making it challenging to schedule interviews or obtain their input in a timely manner.
- **Limited User Representation:** It's crucial to ensure that a diverse range of users is represented during data gathering. However, it is challenging to reach and include a wide variety of potential users, leading to a potential bias in the gathered data.
- **Resistance to Change:** Some stakeholders, especially restaurant owners or employees, resistant to change or skeptical about adopting new technology. They are reluctant to share their concerns and are not fully cooperative in the data collection process.

Remedies for obstacles

- Limited Stakeholder Availability: Plan ahead and schedule interviews or meetings well in advance to accommodate stakeholders' busy schedules.
- Offer flexible meeting options, such as online video conferences, to make it easier for stakeholders to participate.
- Provide clear and concise communication about the purpose, importance, and expected duration of the data gathering process to motivate stakeholders to prioritize their involvement.

- Limited User Representation: Employ various recruitment strategies to ensure a diverse range of users are included, such as targeting different demographics or conducting surveys at various locations.
- Consider conducting user testing sessions with a representative sample to gather feedback on the system's usability and functionality.

- Resistance to Change: Clearly communicate the benefits and advantages of the online food ordering system to stakeholders, emphasizing how it can streamline operations, increase efficiency, and improve customer satisfaction.
- Address concerns and objections proactively by providing examples or case studies of successful implementations of similar systems in the industry.

3.2 Analysis of the gathered data

Based on the data collected through market research, document analysis, interviews, and observations, the analysis highlights the need for a food ordering system with a mobile application to enable efficient order management and tracking for delivery person. The gathered data reveals several key findings and requirements, as described below:

Growing Demand for Food Delivery Services: Market research indicates a significant increase in the demand for online food delivery services. Customers are increasingly relying on online platforms to order food from restaurants, indicating a need for a streamlined system that facilitates the ordering and delivery process.

Improving Delivery Efficiency and Transparency: Observations and interviews with delivery person highlighted the need for a mobile application that provides detailed order information. Such features would enable delivery person to efficiently plan their routes, minimize delivery times, and provide a transparent and seamless experience for customers.

3.3 Extant system

3.3.1 Process of the current system

The online food ordering system enables customers to place orders through a user-friendly web application with a mobile application for delivery person to view order details. Restaurant receive and manage orders, while delivery person track orders and ensure timely delivery. The system streamlines the entire ordering and delivery process, enhancing efficiency, and customer satisfaction.

3.3.2 Problems and limitations of the extant system

- **Absence of Real-Time Order Tracking:** The system lacks real-time order tracking for customers. This means that customers are unable to track their orders in real-time to know the exact status and estimated delivery time. This can lead to uncertainty and customer dissatisfaction, especially when there are delays or unexpected changes in the delivery schedule.
- **Lack of Customer Ratings and Feedback:** The system does not allow customers to give ratings and feedback for their orders. This lack of customer ratings and feedback affects the system's ability to leverage valuable input for improving service quality and addressing customer concerns
- **Inefficient Restaurant Order Management:** The system does not provide a clear and efficient dashboard for restaurant management to handle incoming orders effectively. Without a centralized and organized view of orders, restaurants may face difficulties in managing order queues, assigning orders to specific staff members, and ensuring timely order preparation. This can result in delays, miscommunications, and errors in fulfilling customer orders.
- **Limited Order Customization Options:** The system does not provide extensive customization options for customers to personalize their orders based on dietary restrictions, allergies, or specific preferences. This limitation can be a drawback for customers who require special meal preparations or have specific dietary requirements, potentially leading to dissatisfaction and a reduced customer base.

3.4. Requirement Specification

3.4.1 Functional Requirements

User Registration and Authentication:

- Customers able to create an account or log in to the system using their credentials.

- In the mobile application delivery persons have a separate registration and authentication process.
- The system ensures secure authentication and protect user privacy.

Menu Browsing and Ordering:

- Customers able to browse through restaurant menus, view item details, and add items to their cart.
- The system provides search and filtering options to help customers find specific items or restaurants.
- The system allows customers to review their order, choose a delivery address, and select a payment method.

Order Management for Restaurants:

- The system displays order details, including customer information, order items, delivery address, and any special instructions.

Delivery Tracking for Delivery Person:

- Delivery person have access to a mobile application that allows them to view assigned orders and track their delivery address.
- The system displays order details, including customer information, delivery address, order items, and any special instructions.

3.4.2 Nonfunctional requirements

Performance:

- The system should handle a high volume of concurrent users and orders without significant slowdowns or performance degradation.
- The response time for loading menus, placing orders, and updating order statuses should be within an acceptable range.
- The system should be able to scale effectively to accommodate increased user traffic during peak hours.

Usability:

- The user interfaces of the web and mobile applications should be intuitive, user-friendly, and easy to navigate.
- The system should provide clear instructions and guidance for customers to place orders and track deliveries.
- The interface for restaurant management should be efficient and provide clear visibility of orders and relevant information.

Security:

- The system should employ secure authentication and encryption mechanisms to protect user credentials, personal information.
- Access controls and permissions should be implemented to ensure that only authorized individuals can view and modify order data.

Reliability:

- The system should have a high level of availability and minimize unplanned downtime or disruptions.
- Measures such as regular backups, fault tolerance, and disaster recovery plans should be in place to ensure data integrity and system continuity.
- The system should handle exceptions and errors gracefully, providing informative error messages and appropriate error handling mechanisms.

Compatibility:

- The web and mobile applications should be compatible with different browsers, operating systems, and device types to cater to a wide range of users.
- The system should integrate seamlessly with third-party services, such as payment gateways, SMS notifications, and mapping/navigation APIs.

4.Chapter 4: Design

4.1. System Model

4.1.1 Use case diagram

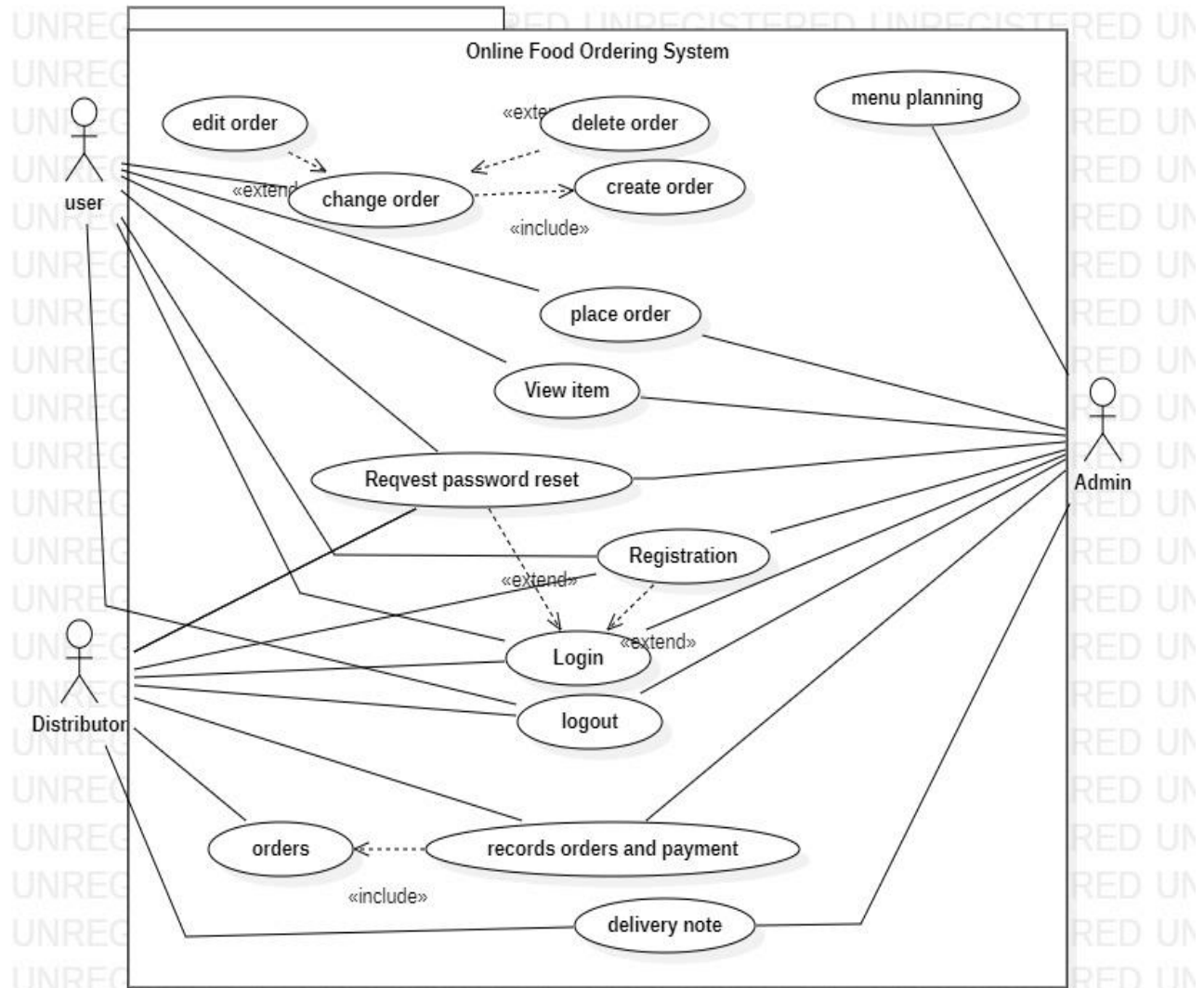


Figure 03- 4.1.1 Use case diagram

4.1.2 Class diagram

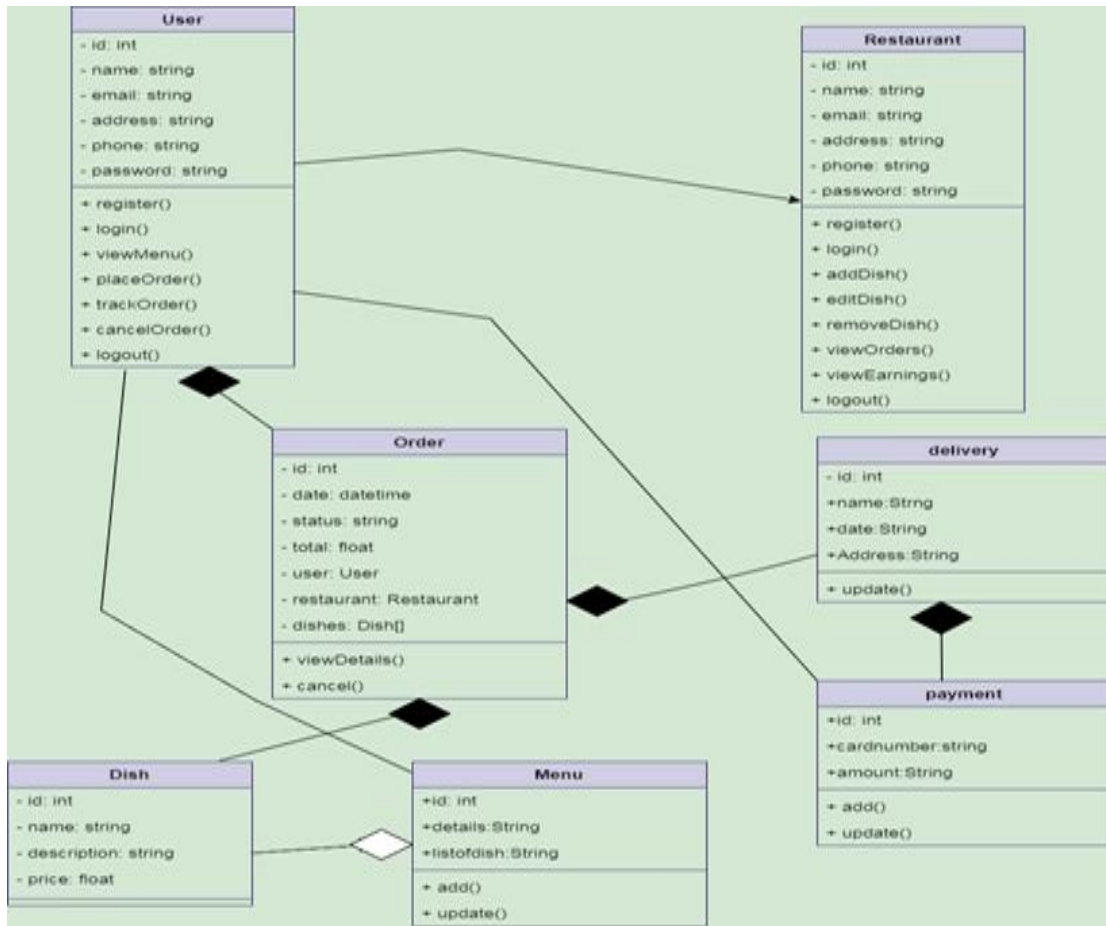


Figure 04- 4.1.2 Class diagram

4.1.3 Activity diagram

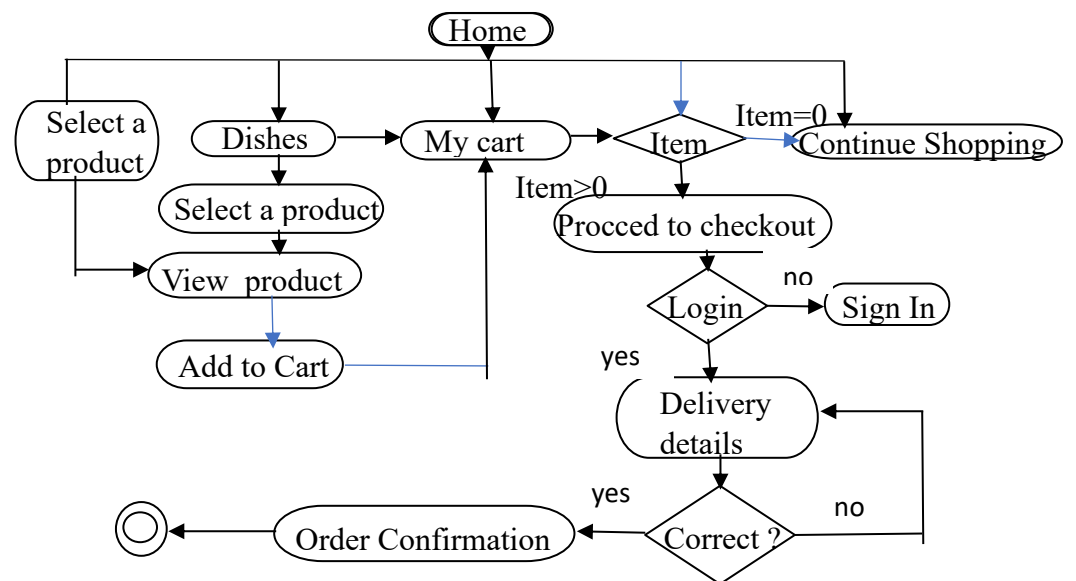


Figure 05- 4.1.3 Activity diagram

4.1.4 Sequence diagrams

Sequence diagram for user registration

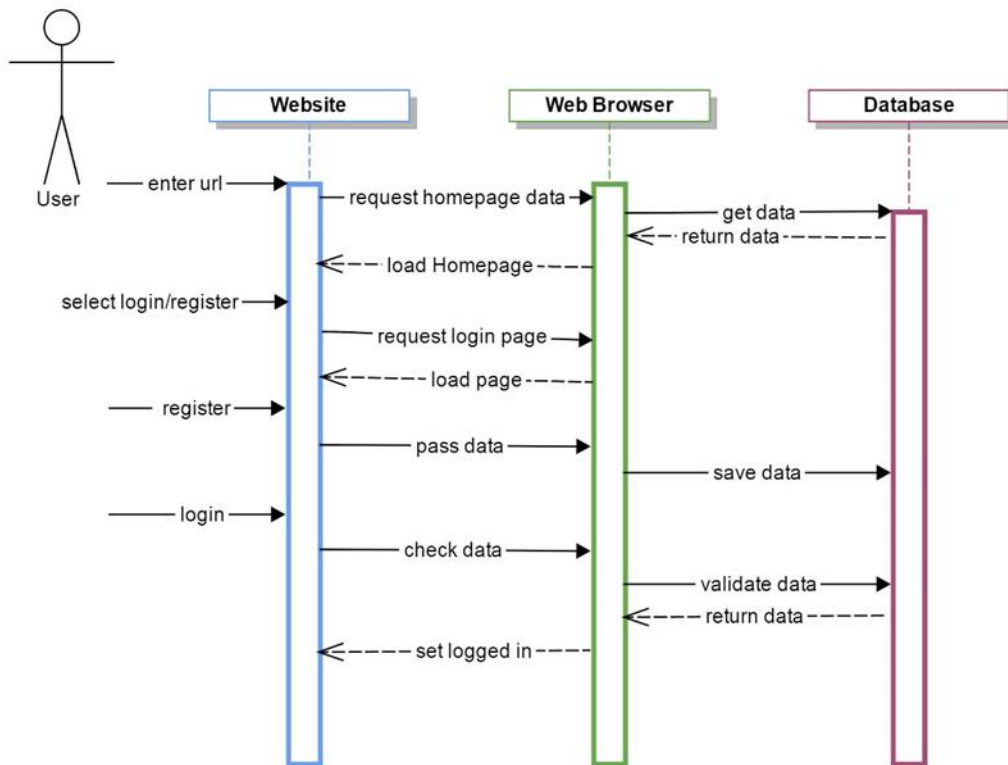


Figure 06- 4.1.4 Sequence diagram for user

Sequence diagram for placing order

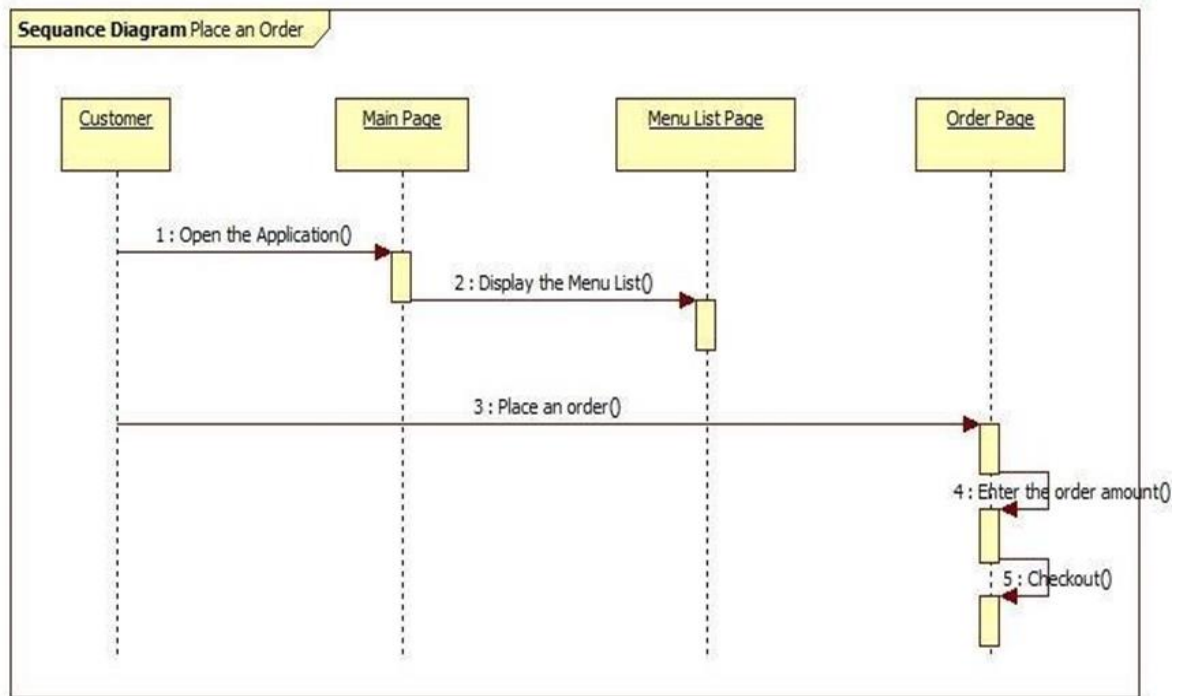


Figure 07-4.1.4 Sequence diagram for placing order

Sequence diagram for delivery

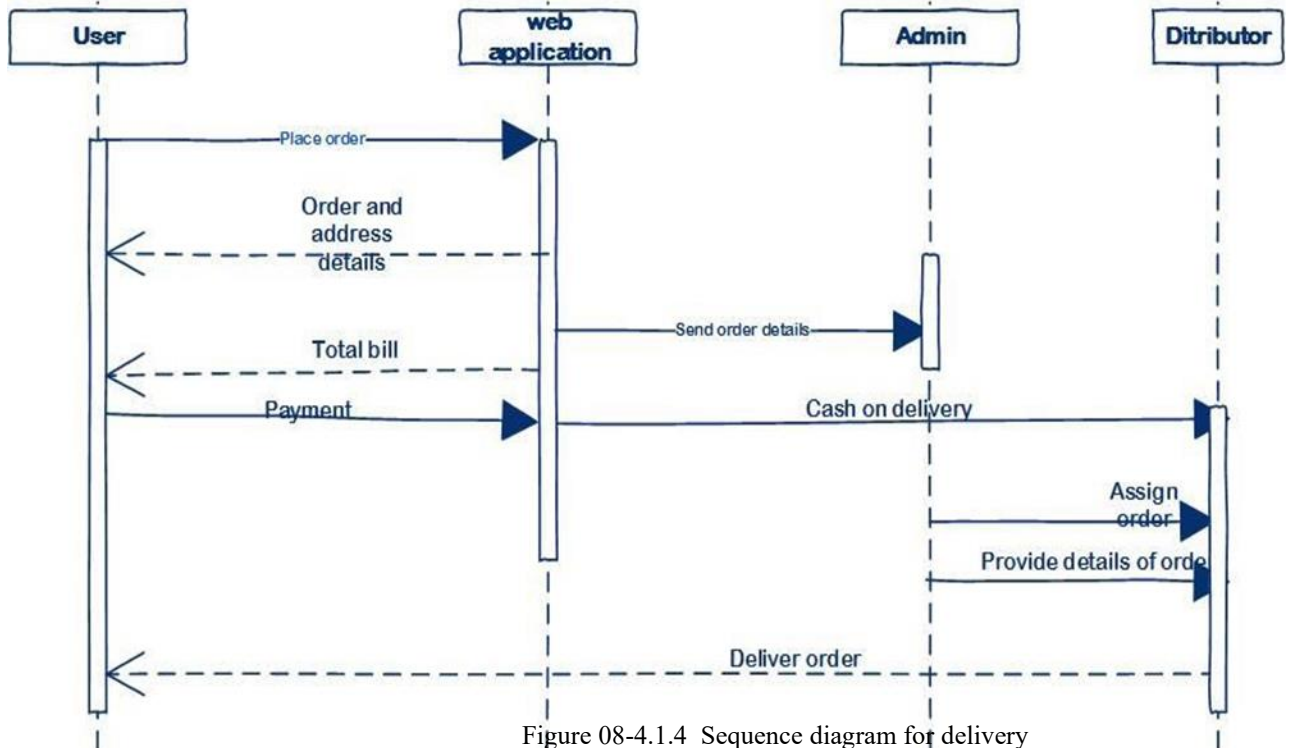


Figure 08-4.1.4 Sequence diagram for delivery

4.2. System Architecture

Web interface level represent presentation layer and Application Layer represent by mobile interface level and Database server level represent database layer in this system.

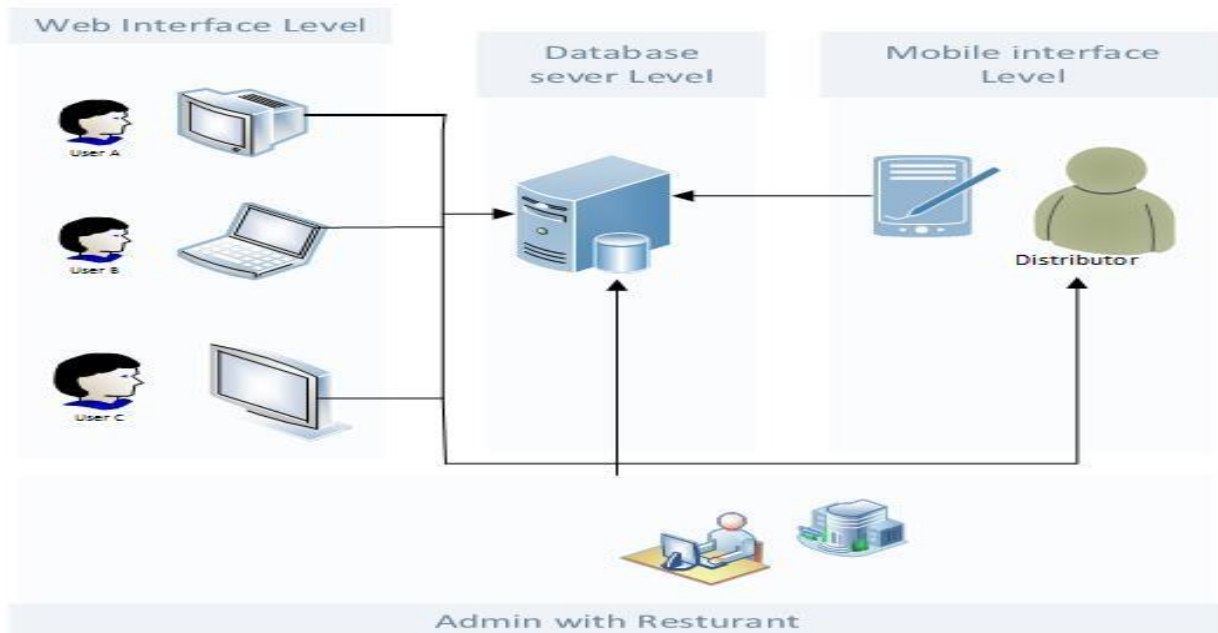
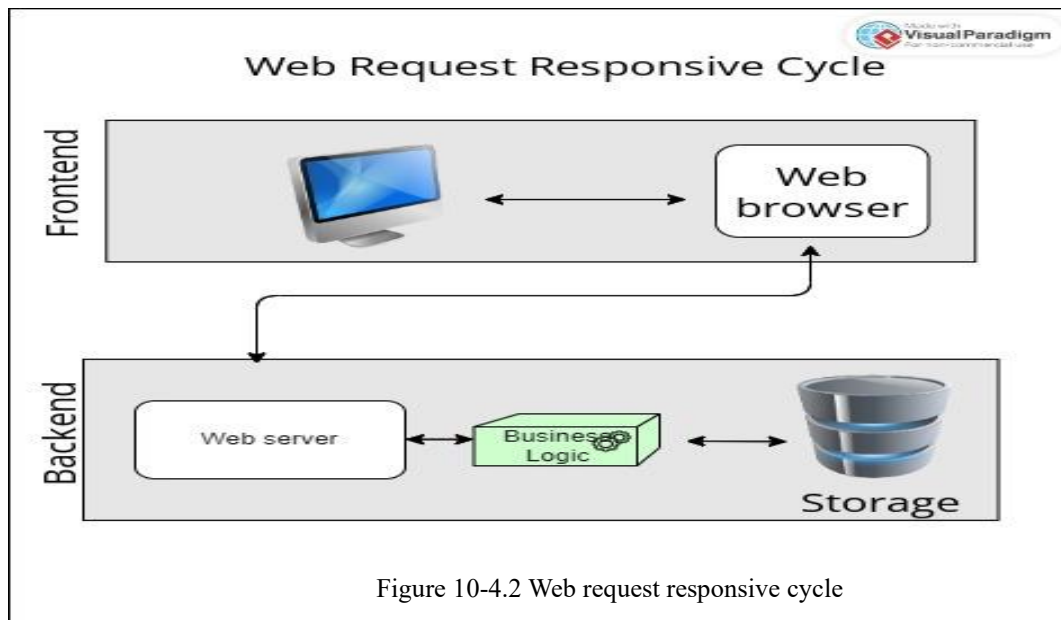
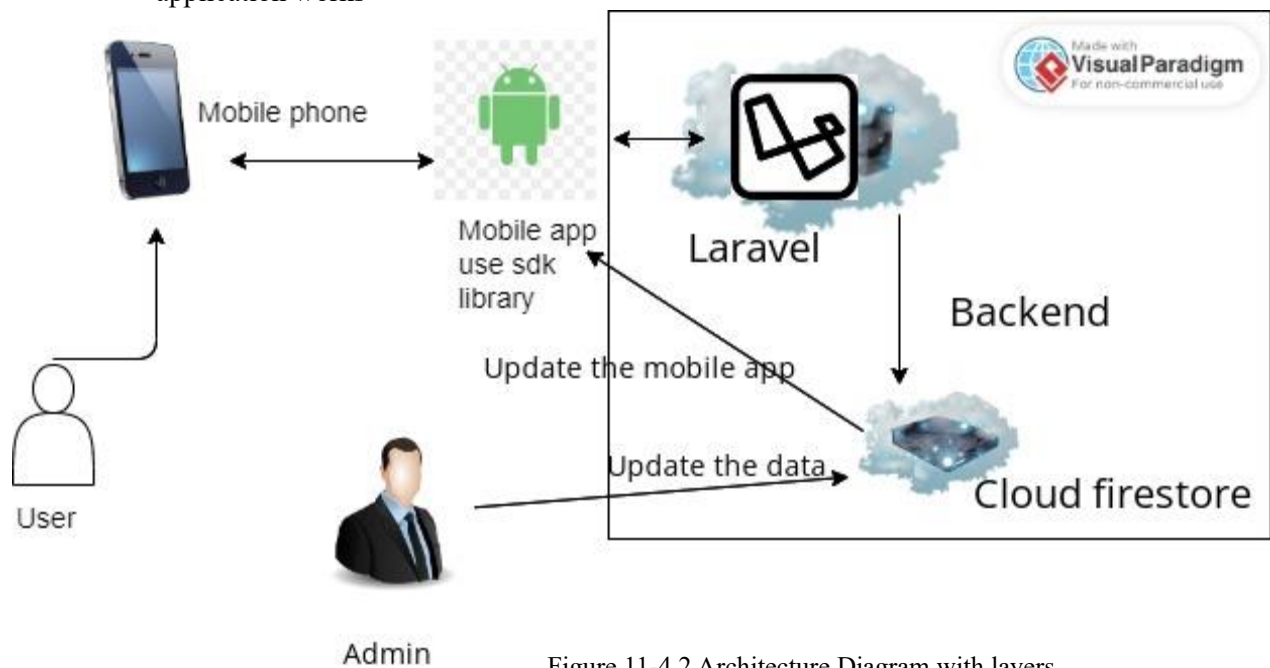


Figure 09- 4.2. System Architecture
Web request responsive cycle



Architecture Diagram This Architecture diagram shows how the infrastructure of this mobile application works



4.3. Database Design

4.3.1. Entity relationship diagram

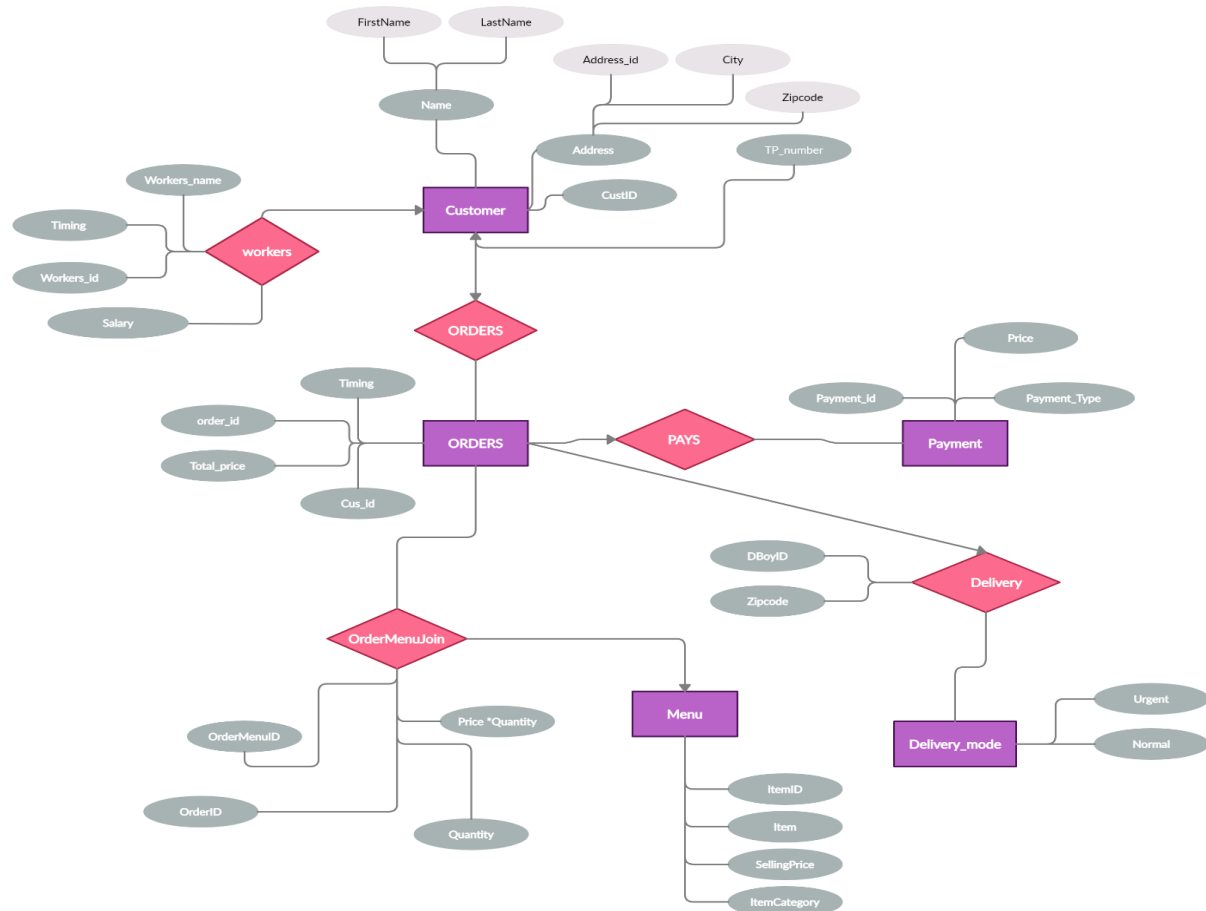


Figure 12- 4.3.1. Entity relationship diagram

4.3.2. Schema diagram

4.4 Interface Design

4.4.1. Considerations when designing interfaces

Here we focused to needs, goals and expectations of users that are going to use the system. We created interface according to the human mental model. Icons images and buttons are used to identify actions easily in interface

Keep the interface consistent to make it easier for people to comprehend and anticipate how various elements will behave. Learnability and usability are improved by using visual design, interaction patterns, and terminology consistently. To maintain consistency across platforms and devices, adhere to established design standards and guidelines. We conduct usability testing with real users to identify issues and effectiveness of system. There are web application interfaces and mobile application interfaces.

4.4.2. Used tools

We built the system with web application and mobile application

Web application-xampp server, html,css,javascript,php,Laravel framework.

Mobile application-Android Studio,Java,Mysql,Emulator

4.4.3. Screen shots of the interface Mobile App Development

➤ Login page

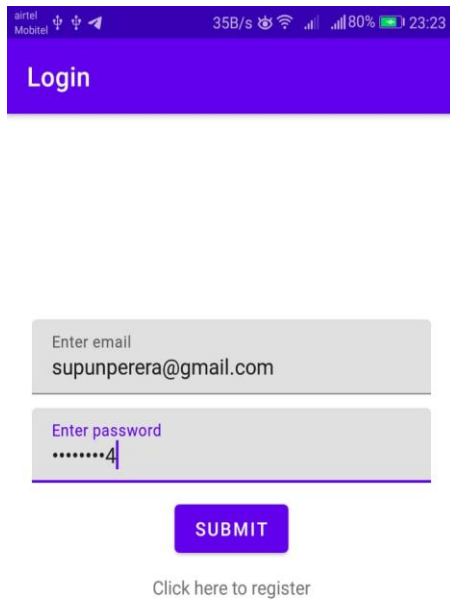


Figure 13- 4.4.3 Login page

Some codes of this page

```
<?php
if(!empty($_POST['email']) && !empty($_POST['password'])) {
    $email=$_POST['email'];
    $password=$_POST['password'];
    $result=array();

    $con = mysqli_connect(hostname:"localhost" , username:"root" , password:"", database:"ecommerce");

    if($con){
        $sql="select * from appusers where email='".$email.'";
        $res=mysqli_query($con,$sql);
```

```

if(mysqli_num_rows($res)!=0){
    $row=mysqli_fetch_assoc($res);
    if($email==$row['email'] && password_verify($password,$row['password'])){
        try{
            $apikey=bin2hex(random_bytes(23));
        }
        catch(Exception $e){
            $apikey=bin2hex(uniqid($email,true));
        }
        $sqlUpdate="update appusers set apikey='".$apikey.'" where email='".$email.'";
        if(mysqli_query($con,$sqlUpdate)){

            //$nme=$row['name'];
            //$ema=$row['email'];

            //$result=array("status"=>"success","message"=>"Login successful");
            //$result=array("status"=>"success","message"=>"Login successful","name"=>$nme);
            //$result=array("status"=>"success","message"=>"Login
successful","name"=>$name,"email"=>$email,"apikey"=>$apiKey);
            $result=array("status"=>"success","message"=>"Login
successful","name"=>$row['name'], "email"=>$row['email'], "apikey" => $apikey);
            //$result=array("status"=>"success","message"=>"Login successful");
            //$result=array("status"=>"success","message"=>"Login
successful","name"=>$row['name'], "email"=>$row['email']);
        }
        else $result=array("status"=>"failed","message"=>"login failed and try again");
    }
    else $result=array("status"=>"failed","message"=>"Retry with correct email and password");
}

```

```

    }
    else $result=array("status"=>"failed","message"=>"Retry with correct email and password");
}
else $result=array("status"=>"failed","message"=>"Database connection failed");
}
else $result=array("status"=>"failed","message"=>"All fields are required");

```

➤ Registration page

<?php Figure 14- 4.4.3 Registration page

```

if(!empty($_POST['name']) && !empty($_POST['email']) && !empty($_POST['password'])) {

    $con = mysqli_connect(hostname:"localhost" , username:"root" , password:"", database:"ecommerce");

    $name=$_POST['name'];

    $email=$_POST['email'];

```

```

$password=password_hash($_POST['password'], algo:PASSWORD_DEFAULT);

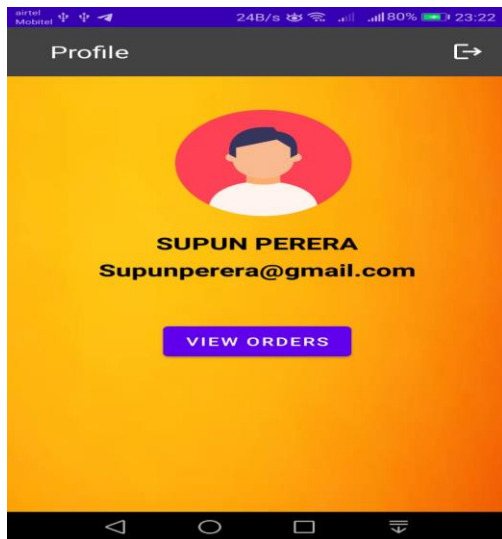
if($con){
    $sql = "insert into appusers (name, email, password) values ('".$name."','".$email."','".$password."')";
    if(mysqli_query($con, $sql)){
        echo "success";
    }
    else echo "Registration failed";
}

else echo "Database connection failed";
}

else echo "All fields are required";

```

➤ Profile of distributor



➤ Order details page

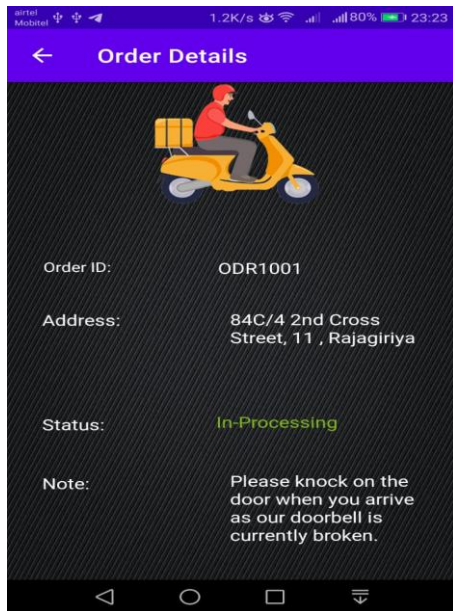


Figure 16- 4.4.3 Order details page

```
<?php
if(!empty($_POST['item'])) {
    $item=$_POST['item'];

    $result=array();

    $con = mysqli_connect(hostname:"localhost" , username:"root" , password:"",
    database:"loginregister");

    if($con){
        $sql="select * from order_details where userID ='".$item."'";
        $res=mysqli_query($con,$sql);

        if(mysqli_num_rows($res)!=0){
            $row=mysqli_fetch_assoc($res);

            //$result=array("status"=>"success","message"=>"Login successful");
            //$result=array("status"=>"success","message"=>"Login successful","name"=>$nme);
        }
    }
}
```

```
// $result=array("status"=>"success","message"=>"Login
successful","name"=>$name,"email"=>$email,"apikey"=>$apiKey);
```

```
$result=array("status"=>"success","message"=>"Login
successful","id"=>$row['userID'], "address"=>$row['adress'], "order_status"=>$row['order_status'], "note"=
>$row['note']);
```

```
// $result=array("status"=>"success","message"=>"Login successful");
```

➤ orders

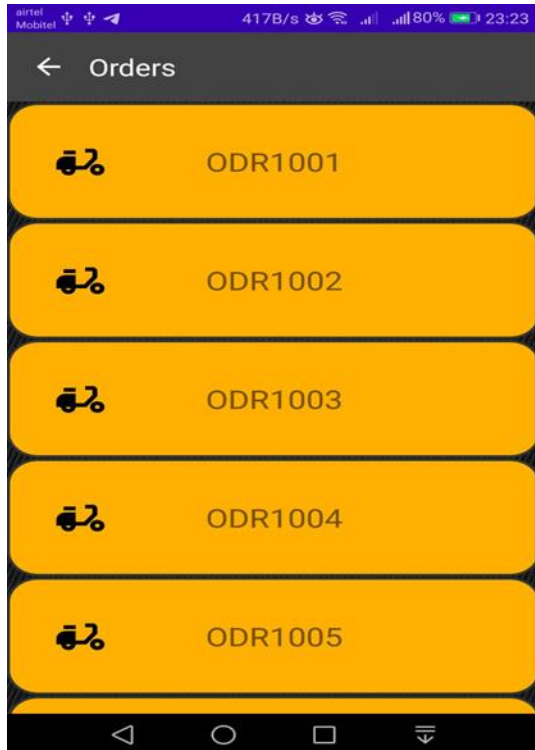


Figure 17-4.4.3 orders

```
public class viewOrderDetails extends AppCompatActivity {
    Button goBack;
    TextView showUserId,showAddress,showOrderStatus,showNote;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_order_details);
        showUserId=findViewById(R.id.textViewUseID);
        showAddress=findViewById(R.id.textViewAddress);
```

```

showOrderStatus=findViewById(R.id.textViewOrderStatus);
showNote=findViewById(R.id.textViewNote);

getSupportActionBar().setTitle("Order Details");

String sessionId = getIntent().getStringExtra("EXTRA_SESSION_ID");
showUserId.setText(sessionId);
String item=sessionId;

RequestQueue queue = Volley.newRequestQueue(getApplicationContext());
String url ="http://172.31.98.165/FoodOrder/orderdetails.php";

StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
    new Response.Listener<String>() {

        @Override
        public void onResponse(String response) {

            try {
                JSONObject jsonObject=new JSONObject(response);
                String orderId=jsonObject.getString("id");
                String address=jsonObject.getString("address");
                String orderStatus=jsonObject.getString("order_status");
                String note=jsonObject.getString("note");

                System.out.println(orderId);
                System.out.println(address);
                System.out.println(orderStatus);
                System.out.println(note);
            }
        }
    }
);

```

```
showUserId.setText(orderId);  
showAddress.setText(address);  
if(orderStatus.equals("0")){  
    showOrderStatus.setText("In-Processing");  
}  
else{  
    showOrderStatus.setText("Delivering");  
}  
showNote.setText(note);
```

Web development

➤ Home page

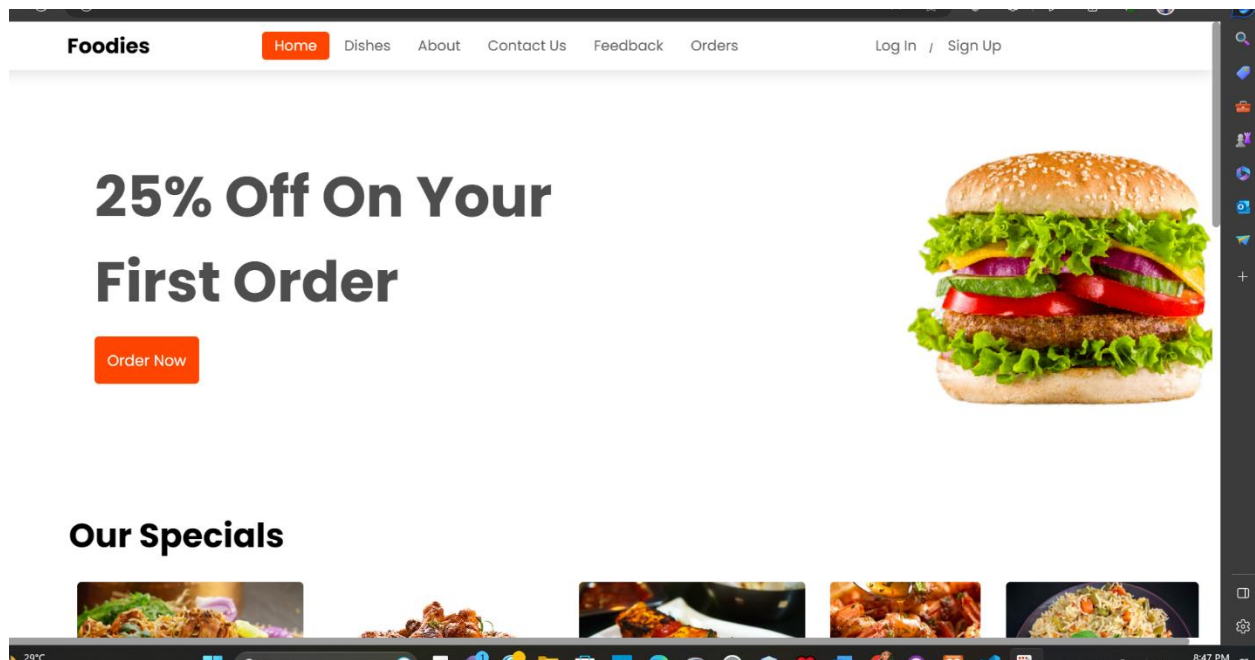


Figure 18-4.4.3 Home page of web application-1

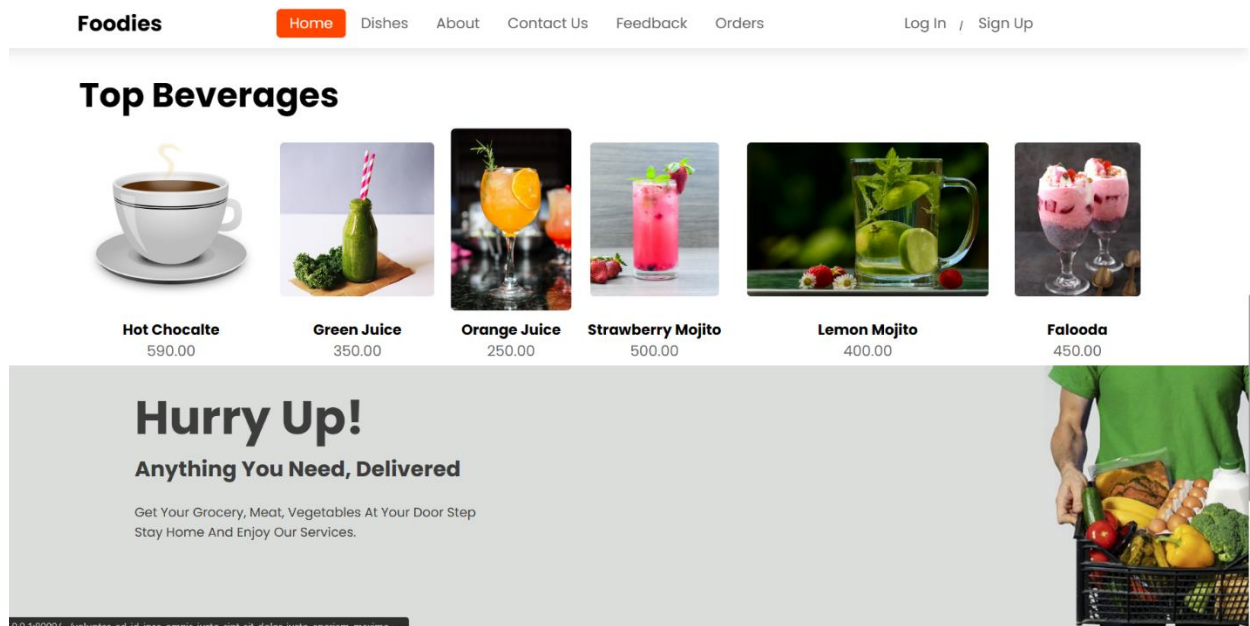


Figure 19-4.4.3 Home page of web application-2

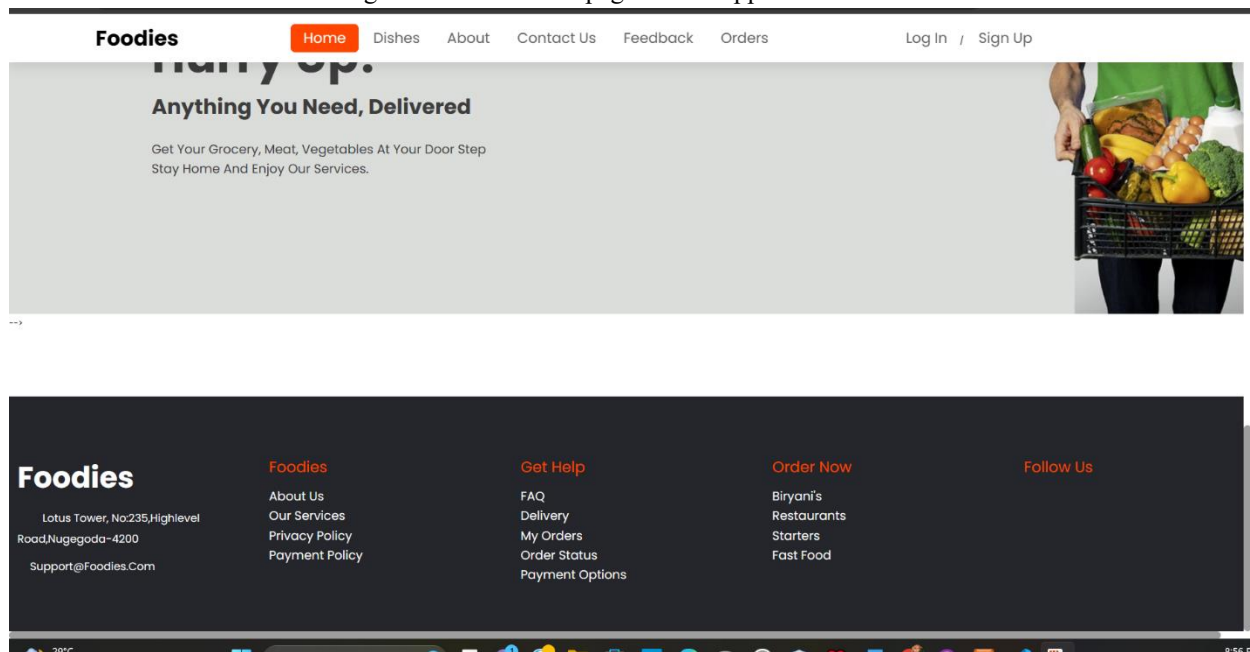


Figure 20-4.4.3 Home page of web application-3

```
<!--Home section start-->
<section class="home" id="home-section">
<!--Add-->
<div class="add">
  <div class="add-container">
    
    <div class="textimg"><h2>25% off on your<br> first order</h2>
    <a href="{ { route('dish') } }"><button class="ordr">Order Now</button></a></div>
```

```

</div>
</div>
<!--Add ends-->
<!--Our-Specials-->

<div class="spl">
    <h2>Our Specials</h2>

</div>
<div class="table">
    @foreach($foods as $food)
        <a href="{{ route('foods.details',['slug'=>$food->slug]) }}">
            name }}" height="150">
            <h4>{{ $food -> name }}</h4>
            <p>{{ $food->regular_price }}</p></a>

    @endforeach

</div>
<div class="top-re">
    <h2>Top Beverages</h2>
</div>
<div class="table1">
    @foreach($beverages as $beverage)
        <a href="{{ route('beveragedetail',['slug'=>$beverage->slug]) }}">
            name }}" height="150">
            <h4>{{ $beverage -> name }}</h4><p>{{ $beverage->regular_price }}</p>
            </a>
    @endforeach

</div>
<div class="container-grocery">
    
    <div class="text-block">
        <h2>Hurry Up!</h2>
        <h4>Anything you need, delivered</h4>
        <p>Get your grocery, meat, vegetables at your door step<br>
        Stay Home and Enjoy our services.</p>
    <!--<div class="ordergrocery">

```

Home component livewire file

```
<?php

namespace App\Http\Livewire;
use App\Models\foods;
use App\Models\beverages;
use Livewire\Component;
use Livewire\WithPagination;
use Cart;

class HomeComponent extends Component
{
    use WithPagination;
    public function store($product_id,$product_name,$product_price){
        Cart::add($product_id,$product_name,1,$product_price)->associate("\App\Models\foods");
        session()->flash('success_message1','Item added in cart');
        return redirect()->route('addtocart');
    }
    public function render()
    {
        $foods=foods::paginate(6);
        // dd($foods->first());
        $beverages=beverages::paginate(6);
        //dd($beverages->first());
        return view('livewire.home-component',['foods'=>$foods],['beverages'=>$beverages]);
    }
}
```

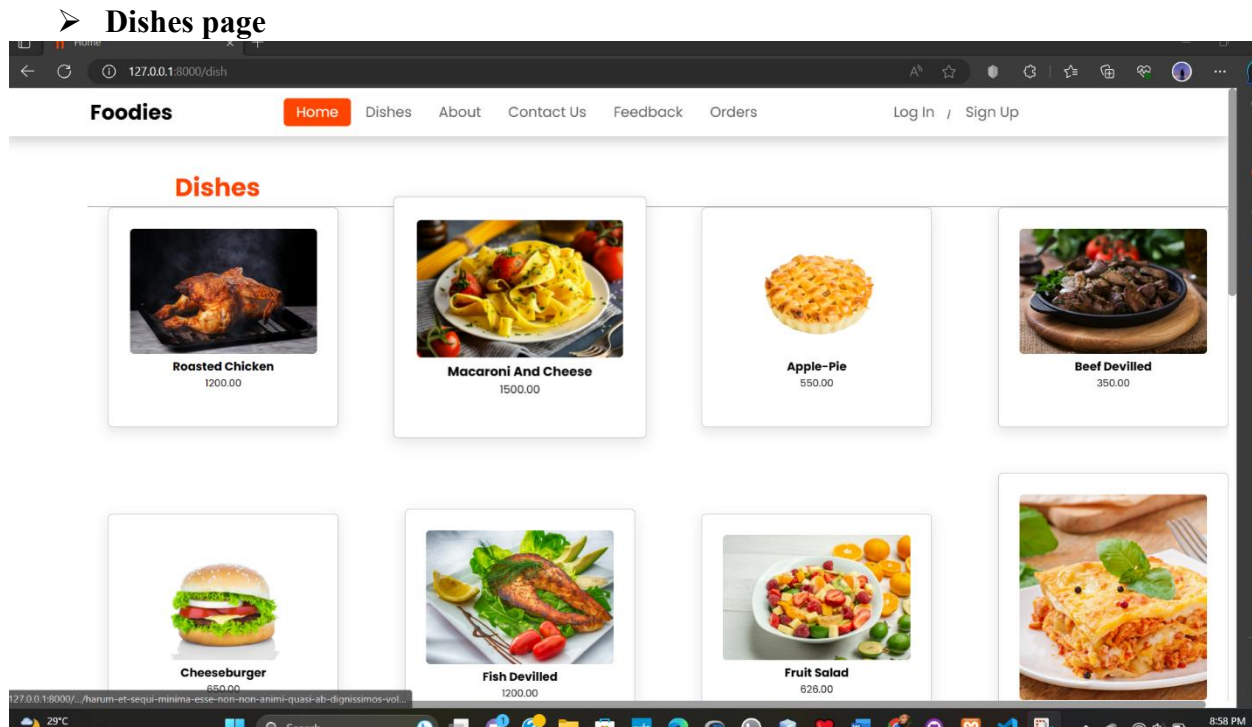


Figure 21-4.4.3 Dish page of web application

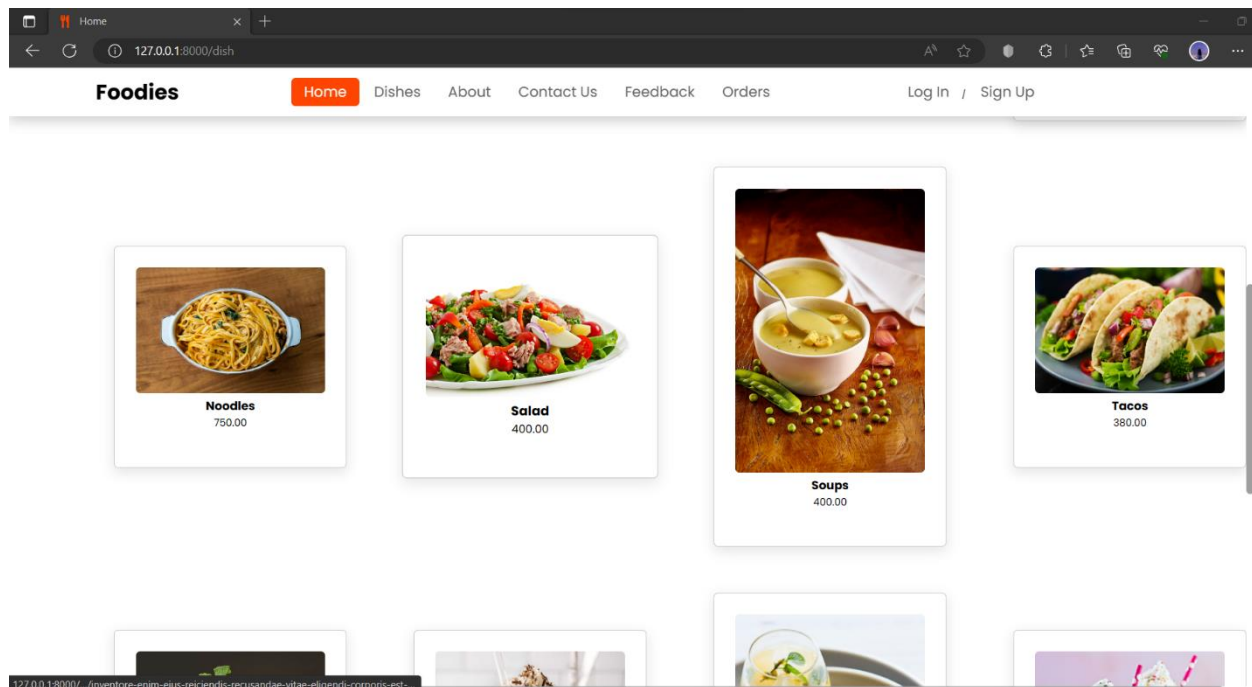


Figure 22-4.4.3 Dish page of web application

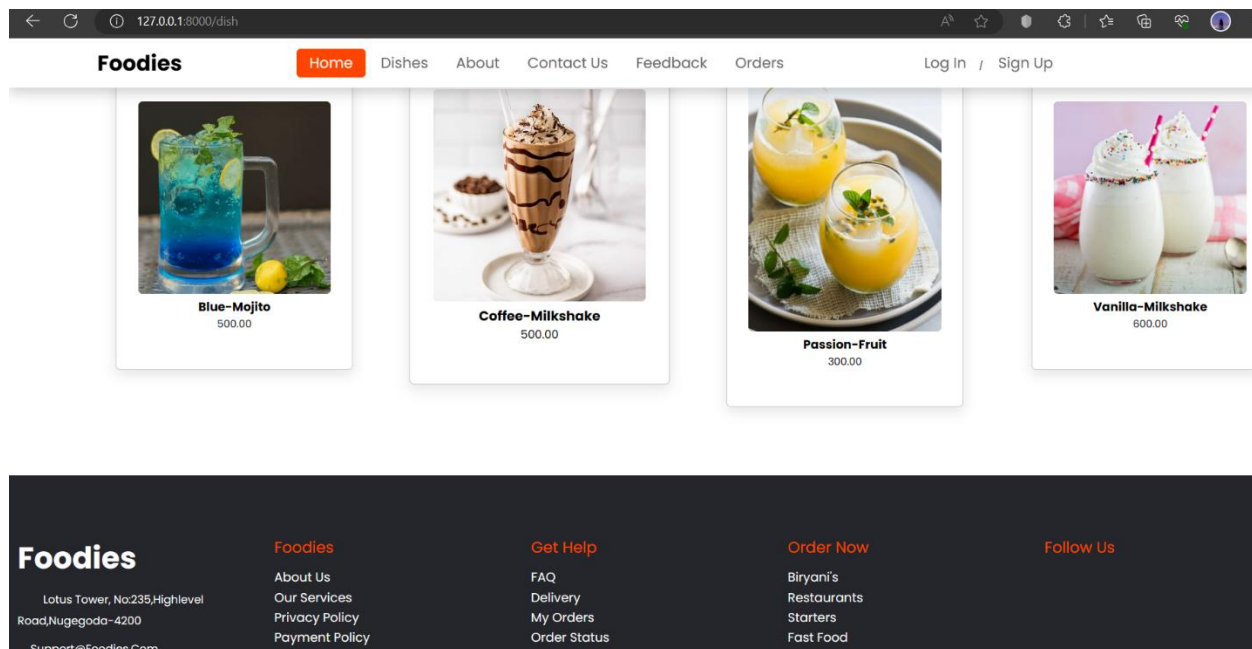


Figure 23-4.4.3 Dish page of web application

dish component file

```
<body>
  <!--Header section start-->
  <header>
    <a href="#" class="logo"><i class="fa fa-utensils"></i>Foodies.</a>
    <nav class="navbar">
      <a href="{ {route('home.index')}}">Home</a>
      <a class="active" href="{ {route('dish')}}">dishes</a>
      <a href="#" onclick="openAbout()">about</a>
      <a class="feed" id="feedback">feedback</a>
      <a href="{ {route('contact')}}">Contact us</a>
      <a href="{ {route('addtocart1')}}">Orders</a>
    </nav>
    <div class="icons">
      <i class="fas fa-bars" id="menu-bars"></i>
      <i class="fas fa-search" id="search-icon"></i>
      <a href="#" class="fas fa-heart"></a>
      <a href="#" class="fas fa-shopping-cart"></a>
      <i class="fa fa-user" aria-hidden="true"></i>
      <a href="login.html" class="fas fa-sign-in-alt"></a>
    </div>

    <!--search form-->
    <form action="" id="search-form">
```

```

<input type="search" placeholder="search here..." name="" id="search-box">

<section>
  <h3>Dishes</h3>
</section>
<section class="barb" id="biryani">
  <h1 class="barbeque">Dishes</h1>
  <hr class="line">
  <div class="box-container">
    @foreach($Post as $Posts)
    <div class="box">
      <a href="{ { route('dishdetail',['slug'=>$Posts->slug]) }}"></a>
      <h3>{ { $Posts->name } }</h3>
      <p>{ { $Posts->regular_price } }</p>
      <div class="stars">
        <i class="fas fa-star"></i>
        <i class="fas fa-star"></i>
        <i class="fas fa-star"></i>
        <i class="fas fa-star"></i>
        <i class="fas fa-star-half-alt"></i>
      <br>

```

Dish component livewire file

```

<?php

namespace App\Http\Livewire;

use Livewire\Component;
use Cart;
use App\Models\Post;
use Livewire\WithPagination;

class DishComponent extends Component
{
    use WithPagination;
    public function render()
    {
        $Post=Post::paginate(16);
        // dd($Post->first());
        return view('livewire.dish-component',['Post'=>$Post]);
    }
}

```

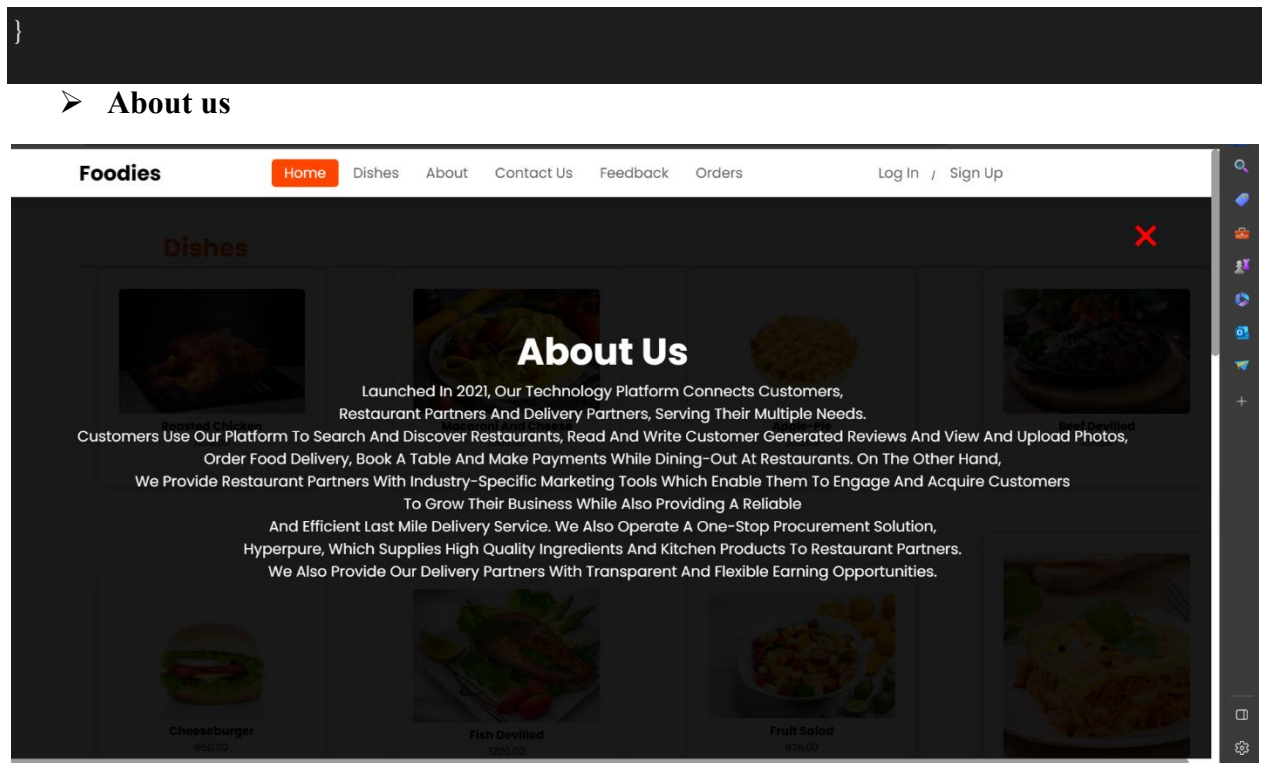


Figure 24-4.4.3 About us page of web application

```

<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <title>{{ config('app.name', 'Laravel') }}</title>

    <!-- Fonts -->
    <link rel="preconnect" href="https://fonts.bunny.net">
    <link href="https://fonts.bunny.net/css?family=figtree:400,500,600&display=swap" rel="stylesheet" />

    <!-- Scripts -->
    @vite(['resources/css/app.css', 'resources/js/app.js'])
  </head>
  <body class="font-sans text-gray-900 antialiased">
    <div class="min-h-screen flex flex-col sm:justify-center items-center pt-6 sm:pt-0 bg-gray-100">
      <div>
        <a href="/">
          <x-application-logo class="w-20 h-20 fill-current text-gray-500" />
        </a>
      </div>
    </div>
  </body>
</html>

```



```

<div class="w-full sm:max-w-md mt-6 px-6 py-4 bg-white shadow-md overflow-hidden sm:rounded-
lg">
  {{ $slot }}
</div>
</div>

```

➤ Contact us



Contact Us

Name:

Figure 25-4.4.3 Contact Us page of web application

```

<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact Us | Foodies</title>
    <link rel="stylesheet" href="contact.css">
    <link rel="icon" href="/Images/Restaurants/download.png" type="image/icon type">
    <script src="https://kit.fontawesome.com/a076d05399.js"></script>
  </head>
  <body>

    <div class="row">
      <div class="box">
        <h2><i class="fas fa-phone"></i> Contact:0112525169</h2>
        <br>
        <h2><i class="fas fa-at"></i> Email: support@foodies.com</h2>
        <br>
        <h2><i class="fa fa-map-marker"></i>Lotus Tower,<br>235,Highlevel Road ,<br>Nugegoda-4200 </h2>
      </div>
    </div>

```



```

</div>
<div class="contact">
<h1>Contact Us</h1>

<form class="form" action="/home.html" id="form">
<label for="uname">Name:</label>
<input type="text" class="uname" name="name" placeholder="Enter Your Full Name">

<br>

```

➤ Login

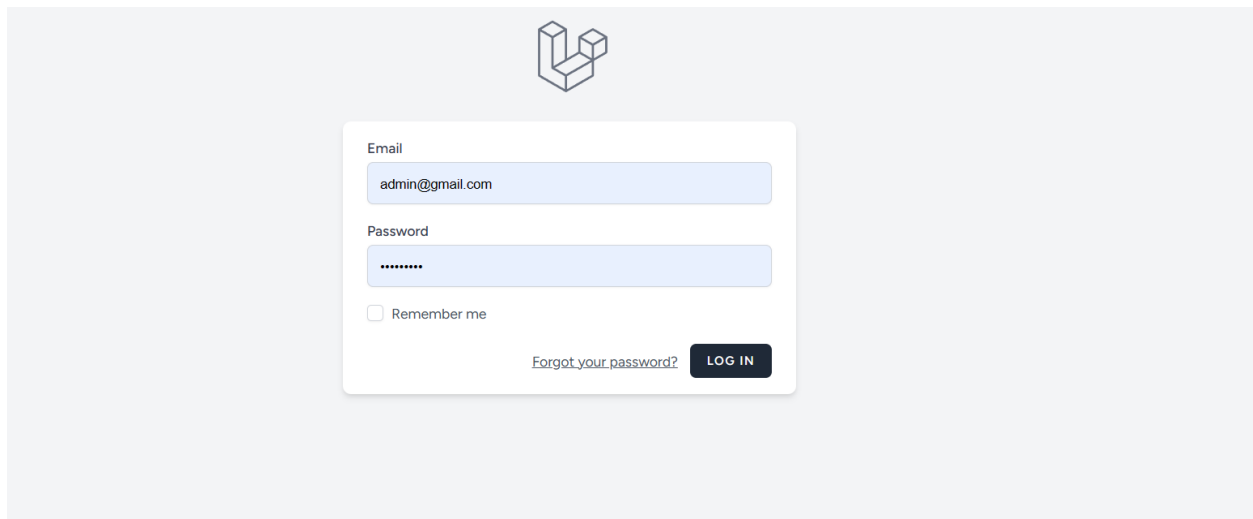


Figure 26-4.4.3 Login page of web application

```

@livewireStyles
</head>
<body onload="myload()">

{{ $slot }}

<!--Header section start-->
<header>
  <a href="#" class="logo"><i class="fa fa-utensils"></i> Foodies</a>

  <nav class="navbar">
    <a class="active" href="{{ route('home.index') }}">Home</a>
    <a href="{{ route('dish') }}">dishes</a>
    <a href="#" onclick="openAbout()">about</a>
    <a href="{{ route('contact') }}">Contact us</a>
    <a class="back" id="feedback">feedback</a>
    <a href="{{ route('addtocart1') }}">Orders</a>
  </nav>

```

```

@auth
<font color="#ff4500"> <h2><b>{{ Auth::user()->name }} / </h2></b></font>
<form method="POST" action="{{ route('logout') }}">
  @csrf
  <font color="#ff4500"> <h2><b>  <a href="{{ route('logout') }}" onclick="event.preventDefault();
this.closest('form').submit();">Log Out</a></form></h2></b></font>
  @else
  <nav class="navbar">  <a href="{{ route('login') }}">Log In</a>/ <a href="{{ route('register') }}">Sign
Up</a></nav>
  @endif

@auth
  @if(Auth::user()->utype == 'ADM')
    <font color="#ff4500"> <h2><b>  <li><a
href="{{ route('admin.dashboard') }}">Dashboard</a></li></h2></b></font>
  @else
    <font color="#ff4500"> <h2><b>  <li><a
href="{{ route('user.dashboard') }}">Dashboard</a></li></h2></b></font>
  @endif
@endif

```

➤ Signup

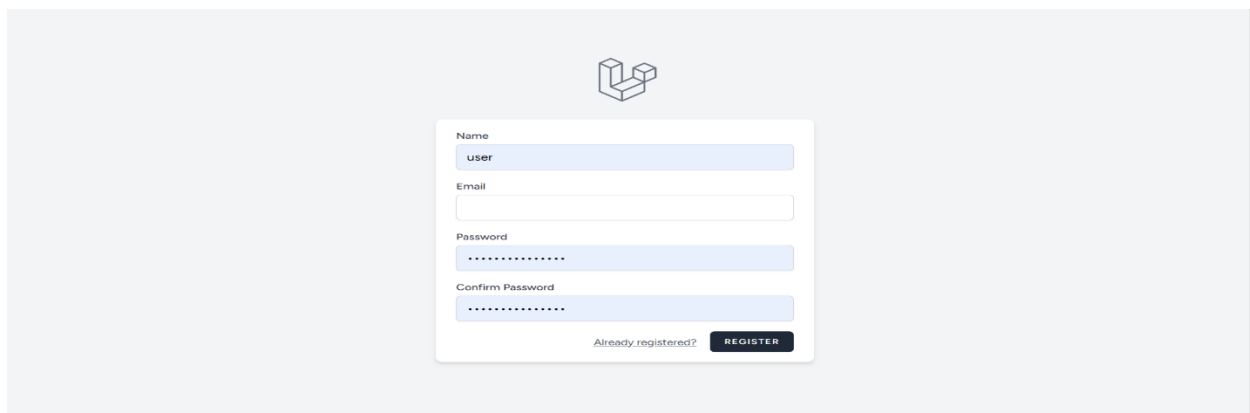


Figure 27-4.4.3 Signup page of web application

➤ Add To Cart

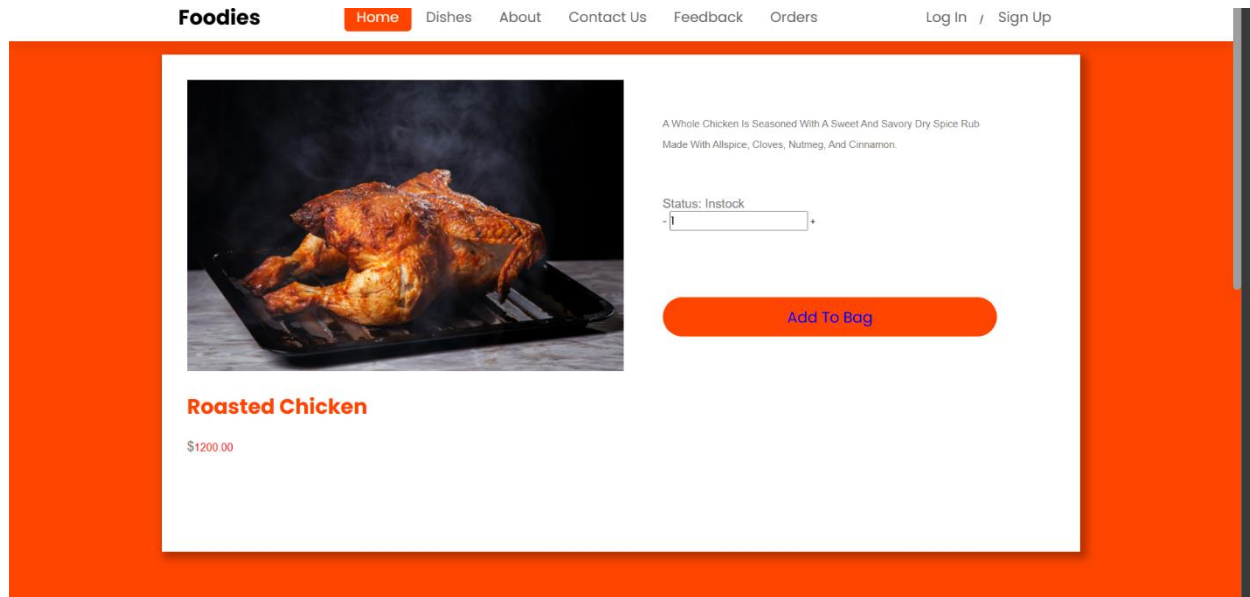


Figure 28-4.4.3 Add to cart page of web application

```
<section>
<div class="box-container">

  <div class="box">
    <div class="container flex">
      <div class="left">
        <div class="main_image">
          
          <h3>{{ $Post->name }}</h3>
          <h4><small>$</small>{{ $Post->regular_price }}</h4>
        </div>

      </div>
      <div class="right">

        <p>{{ $Post->short_description }}</p>
        <h5>Status: {{ $Post->stoke_status }}</h5>
        <div class="counter">
          <span class="down" onClick='decreaseCount(event, this)'>-</span>
          <input type="text" value="1">
          <span class="up" onClick='increaseCount(event, this)'>+</span>
        </div>

      </div>
    </div>
  </div>
</div>
```

```

<script type="text/javascript">
  function increaseCount(a, b) {
    var input = b.previousElementSibling;
    var value = parseInt(input.value, 10);
    value = isNaN(value)? 0 : value;
    value ++;
    input.value = value;
  }
  function decreaseCount(a, b) {
    var input = b.nextElementSibling;
    var value = parseInt(input.value, 10);
    if (value > 1) {
      value = isNaN(value)? 0 : value;
      value --;
      input.value = value;
    }
  }
</script>
  <button><a href="#" wire:click.prevent="store({{$Post->image}},'{{$Post->name}}',{{$Post-
>regular_price}})">Add to Bag</button>

</div> </div>
</div>

</section>
<script>
  function img(anything){
    document.querySelector('.slide').src=anything;
  }
  function change(change){
    const line=document.querySelector('.home');
    line.style.background=chanege;
  }
}
</script>

```

➤ Cart page

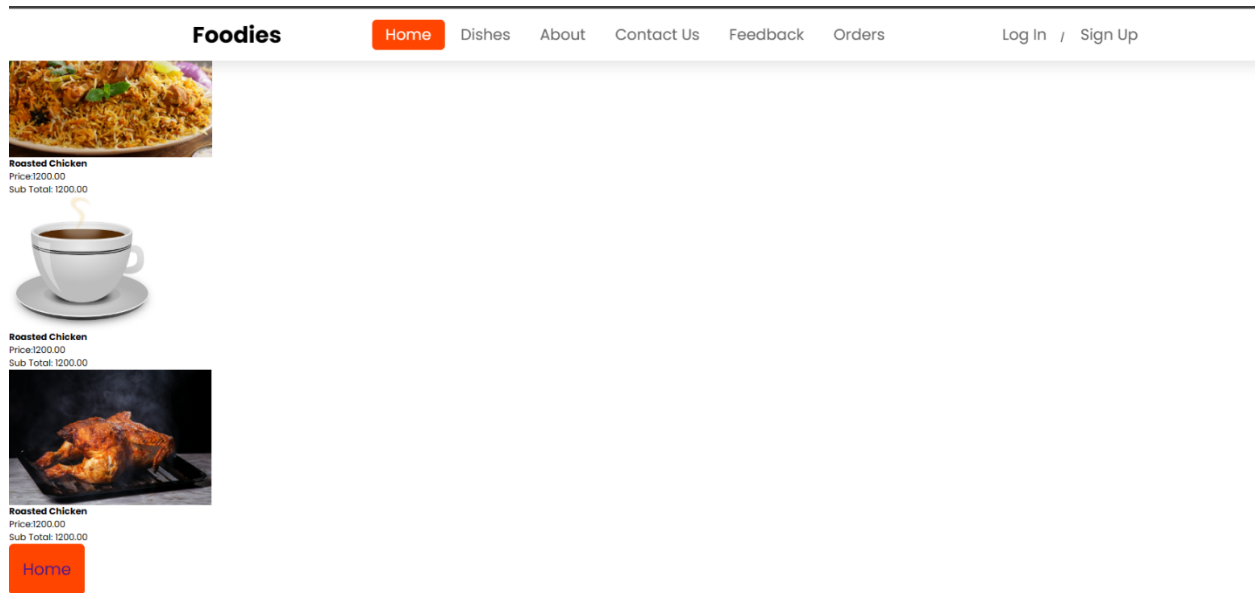


Figure 29-4.4.3 Cart page of web application-1

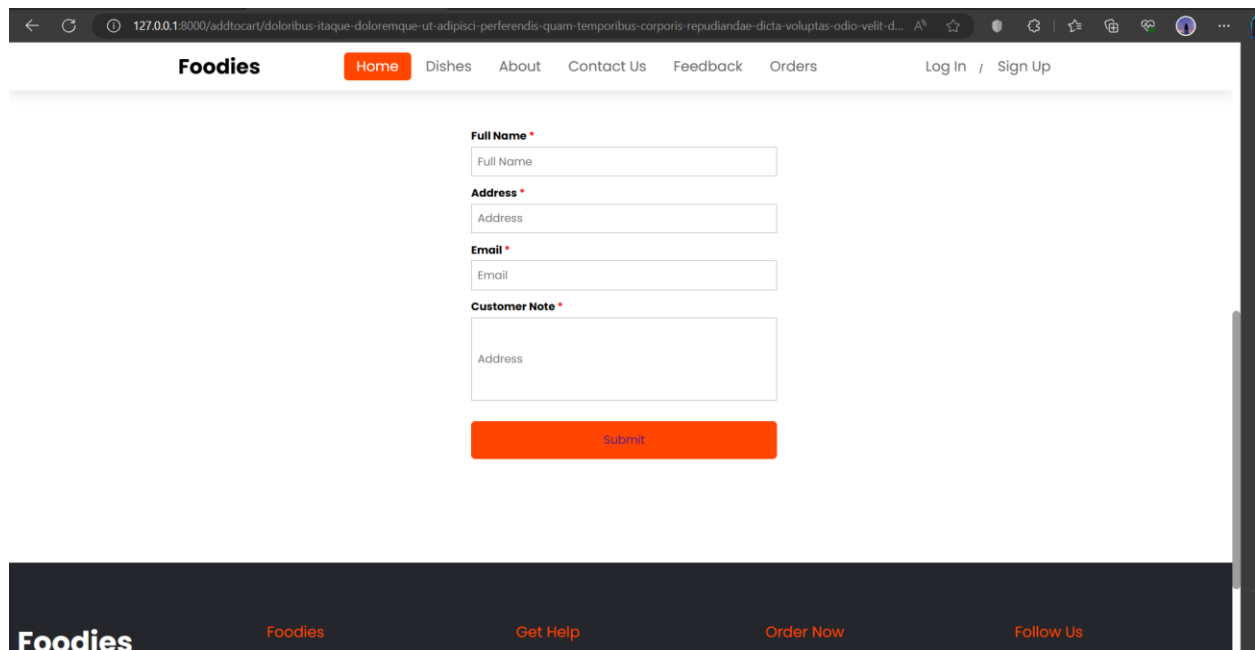


Figure 30-4.4.3 Cart page of web application-2



```

@if(Cart::count()>0)
@foreach(Cart::content() as $item)
    <div class="xy">
        <div class="table">
            model->name }}" height="150"><h4>{{ $item->model-
>name }}</h4><p>Price:{{ $item->model->regular_price }}</p><p>Sub Total: {{ $item->subtotal }}</p>
        </div>
    </div>
@endforeach
@foreach(Cart::content() as $item)
    <div class="xy">
        <div class="table">
            model->name }}" height="150"><h4>{{ $item->model-
>name }}</h4><p>Price:{{ $item->model->regular_price }}</p><p>Sub Total: {{ $item->subtotal }}</p>
        </div>
    </div>
@endforeach
@foreach(Cart::content() as $item)
    <div class="xy">
        <div class="table">
            model->name }}" height="150"><h4>{{ $item->model-
>name }}</h4><p>Price:{{ $item->model->regular_price }}</p><p>Sub Total: {{ $item->subtotal }}</p>
        </div>
    </div>
@endforeach
@if(Session::has('success_message1'))
    <div class="alert alert-success">
        <h4><Strong>Succes |{{ Session::get('success_message1')}}</strong></h4>
    @endif
<button><a href="{{ { route('home.index') }}" >Home</a></button>
    @ if (session()->has('message'))
        <div class="alert alert-success text-center">{{ session('message') }}</div>
    @endif

@else
    <p>no item in the cart</p>
<div class="border-top">
    <ul class="form-style-1">
<form wire:submit.prevent="CustomerD">
    <li>

```

```

    <label>Full Name <span class="required">*</span></label>
    <input type="text" id="name" name="name" wire:model="name" class="field-long" placeholder="Full
Name" />

</li>
<!-- @error('customer_id')
    <span class="text-danger" style="font-size: 12.5px;">{{ $message }}</span>
@enderror -->
<li>
    <label>Address <span class="required">*</span></label>
    <input type="text" id="address" name="address" wire:model="address" class="field-long"
placeholder="Address" />
</li>
<!-- @error('address')
    <span class="text-danger" style="font-size: 12.5px;">{{ $message }}</span>
@enderror -->
<li>
    <label>Email <span class="required">*</span></label>
    <input type="email" id="email" name="email" wire:model="email" class="field-long"
placeholder="Email"/>
</li>
<!-- @error('email')
    <span class="text-danger" style="font-size: 12.5px;">{{ $message }}</span>
@enderror -->
<li>
    <label>Customer Note <span class="required">*</span></label>
    <input type="text" id="note" name="note" wire:model="note" class="field-long field-textarea"
placeholder="Address" />
</li>
<!-- @error('note')
    <span class="text-danger" style="font-size: 12.5px;">{{ $message }}</span>
@enderror -->
<li>
    <button type="submit" class="btn btn-primary btn-sm w-50"><a
href="{{ route('home.index') }}">Submit</button>

```

5. Chapter 5: Development

5.1.Solution

5.1.1.Hardware

Point of Sale (POS) System used to provide the primary interface for restaurant staff to manage incoming orders, update order statuses, and coordinate with the delivery team.

smartphones, tablets, desktop computers, or dedicated self-service kiosks have an intuitive interface to browse menus, select items, specify customization options, and provide.

Printers or Display systems ensure clear communication between the front-of-house and back-of-house staff.

Delivery Management Hardware is used for efficient delivery coordination, hardware components are needed.

The temperature monitoring devices for food safety

5.1.2 Software

Web application(menu selection,placing orders) is application enables restaurant staff ,customers,admin to view and process orders, update menu offerings.

Mobile Application(Delivery Management Software) is efficient delivery coordination, dedicated software. This software allows delivery providers to receive order information, assign drivers, optimize routes, track deliveries in real-time, and update order statuses. It should integrate with mapping and navigation services to facilitate efficient and accurate deliveries.

5.2. Technology adopted

5.2.1.Hardware equipment

- Memory
- Computer
- Mouse

5.2.2.Software tools

- Programming languages-java,php,html,css,Javascript
- Database-Mysql,Xampp

5.2.3.Frameworks

- Laravel
- Bootsrap

5.2.4.Integrated Development Environment (IDE)

- Android studio

6. Chapter 6: Testing and evaluation

6.1. Testing plan

- **Functionality testing:** This involves testing all the functions of the online food ordering system to ensure that they are working as expected. This includes testing features such as user registration, login, menu display, ordering, payment processing, order confirmation, and cancellation.
- **Usability testing:** This involves testing the ease of use and user-friendliness of the online food ordering system. This includes testing the user interface, the layout of menus, the ordering process, and the overall flow of the system. The goal is to ensure that the system is easy to use and intuitive for the end-users.
- **Performance testing:** This involves testing the speed, responsiveness, and scalability of the online food ordering system. This includes testing the system under different traffic loads and stress testing to ensure that it can handle a large number of orders and users at peak times.
- **Security testing:** This involves testing the security of the online food ordering system to ensure that it is secure from unauthorized access, data breaches, and cyber-attacks. This includes testing the authentication and authorization mechanisms, data encryption, and protection against common security vulnerabilities.
- **Compatibility testing:** This involves testing the compatibility of the online food ordering system with different web browsers, devices, and operating systems. This ensures that the system works seamlessly across different platforms and devices.
- **Localization testing:** This involves testing the online food ordering system for localization issues such as language translation, currency conversion, and cultural differences. This ensures that the system is accessible and usable by users from different regions and languages.
- **User acceptance testing:** This involves testing the online food ordering system with actual end-users to evaluate their overall experience and satisfaction with the system. This feedback can be used to improve the system further and make it more user-friendly and effective.

By following this testing plan, you can ensure that your online food ordering system is reliable, user-friendly, secure, and meets the needs and expectations of your customers.

6.2. Testing

6.2.1. Unit testing

This involves testing individual units of code in isolation to ensure that they are functioning correctly. For example, testing the code that handles user registration, login, menu display, and order placement.

6.2.2. Integration testing

This involves testing how different units of code work together when integrated. For example, testing how the user registration code works with the user login code, how the menu display code works with the order placement code, and so on.

6.2.3. System testing

This involves testing the entire system as a whole to ensure that all the components are working together correctly. For example, testing the end-to-end flow of the system from user registration to order confirmation and delivery.

6.2.4. Acceptance testing

This involves testing the system with actual users to ensure that it meets their needs and requirements. For example, testing that users can easily navigate the system, place orders, and receive their food in a timely manner.

Here are some specific examples of tests for each type of testing:

➤ Unit testing:

Test that the user registration form correctly validates user input.

Test that the menu items are displayed correctly with the correct prices and descriptions.

Test that the payment processing code correctly calculates the total cost of an order.

➤ Integration testing:

Test that the user registration code works correctly with the login code.

Test that the menu display code works correctly with the order placement code.

Test that the payment processing code works correctly with the order confirmation code.

➤ **System testing:**

Test the entire end-to-end flow of the system, from user registration to order confirmation and delivery.

Test that all components of the system are working together correctly, including the user interface, the database, and the payment processing system.

Test that the system can handle a high volume of orders without any performance issues.

➤ **Acceptance testing:**

Test the system with actual users to ensure that it meets their needs and expectations.

Test that users can easily navigate the system, place orders, and receive their food in a timely manner.

Gather feedback from users and use it to improve the system further.

By performing these tests, you can ensure that your online food ordering system is reliable, user-friendly, and meets the needs of your customers.

6.3. Test results and conclusions of testing

the test results and conclusions of testing for an online food ordering system would provide insights into the system's functionality, usability, performance, and security. The test results would identify any defects or bugs in the system and provide recommendations for their resolution.

The conclusions would summarize the overall quality of the system and provide an assessment of whether it meets the requirements and objectives of the project. If the system passes all tests, the conclusion would be that it is ready for release or deployment. If there are outstanding issues or defects, the conclusion would be that further testing or development is required before the system can be released or deployed.

The test results and conclusions would also inform future testing and development efforts, such as identifying areas of the system that require improvement or refinement, and providing feedback for the development team to enhance the system's functionality, usability, performance, and security.

7. Chapter 7: conclusion

7.1. conclusion of the project

As a result, the suggested system's conclusion is based on user needs and is user-centered. All user-related difficulties pertaining to every system user were taken into consideration when developing the system. Anyone who is familiar with using an Android smart phone can use this. By giving them a full-edged system, many problems with the mess/tiffin service will be resolved. Thus, the implementation of an online food ordering system is carried out to assist and address one of the major issues facing people. Based on the findings of this study, it can be said that it makes it easier for customers to place orders and provides them with the information they need to do so.

The food website application designed for employees can assist them in taking orders and updating data. It is also designed for administrators so that they may use it to manage the entire food system. A meal menu online can be set up with the use of an online meal ordering system, and clients can place orders with ease. A meal menu online also makes it simple to track orders, maintain customer databases, and enhance food delivery services. Even the messiest person can easily upload photographs and personalize the online menu. Potential clients can quickly access a menu online and place an order whenever it's convenient for them. This results in the presentation of an automated meal ordering system with wireless communication and feedback features. The suggested solution would increase the ordering and billing divisions' efficiency and draw in more customers. Because a lot of people are moving to different cities, the proposed system's scope is justifiable and it can be used by a wide range of people.

7.2. Lessons learned and Skills earned

Lessons learned

- Importance of thorough planning and requirements gathering before starting development
- The significance of frequent testing and continuous integration to catch bugs and errors early on in the development process
- The value of incorporating user feedback and making changes accordingly
- The importance of keeping security and privacy in mind throughout the development process

- The significance of scalability and performance when designing and developing a system that can handle high traffic and usage levels.

Skills earned:

- Proficiency in programming languages and frameworks such as HTML, CSS, JavaScript, and backend technologies like Python, Node.js, or PHP
- Knowledge of relational databases and database management systems such as MySQL or PostgreSQL
- Experience with APIs, RESTful services, and web services
- Understanding of front-end and back-end development and how they interact
- Familiarity with testing frameworks and tools like Selenium, JUnit, or Mocha
- Knowledge of project management methodologies such as Agile or Scrum
- Understanding of user experience (UX) and user interface (UI) design principles.

Overall, developing an online food ordering system requires a broad range of skills and knowledge. By learning from the lessons that emerge during the development process and mastering the skills needed to build a successful system, developers can develop more efficient, secure, and user-friendly systems that meet the needs of their customers.

7.3. Recommendations for improvements

- Improve the user interface (UI) and user experience (UX) design to make the system more user-friendly, intuitive, and engaging for customers.
- Optimize the system for mobile devices, since many customers prefer to use their smartphones to order food online.
- Implement a more efficient and streamlined checkout process to minimize the time and effort required for customers to complete their orders.
- Enhance the system's security and privacy features to protect customer data and prevent security breaches or fraud.

- Provide real-time updates and notifications to customers about the status of their orders, such as when their food is being prepared, cooked, and delivered.
- Integrate the system with social media platforms to allow customers to share their experiences and recommendations with their friends and followers.
- Offer personalized recommendations and promotions to customers based on their order history, preferences, and behavior.
- Provide a feedback mechanism for customers to rate their experience and provide suggestions for improvement.
- Implement a loyalty program to reward customers for their repeat business and encourage them to return.
- Offer multiple payment options to customers, such as credit cards, PayPal, or mobile payments, to increase convenience and accessibility.
- Overall, these improvements can help enhance the system's functionality, usability, performance, and security, and provide a better experience for customers.

Chapter 8: References and Appendixes

8.1. Appendixes

- Online Food Delivery System (HTML/Css)

<https://www.geeksforgeeks.org/design-a-webpage-for-online-food-delivery-system-using-html-and-css/>

<https://www.youtube.com/watch?v=A7qwwmkF13o&t=124s>

- Online Food Ordering System (Laravel)

<https://www.youtube.com/watch?v=UJeaxPvbOTM>

- The Information that are support to our Documentations

<https://dl.ucsc.cmb.ac.lk/jspui/bitstream/123456789/4492/1/2017%20MIT%20034.pdf>

- Architecture design for Online food ordering system

<https://sandesh-deshmane.medium.com/architecture-and-design-principles-for-online-food-delivery-system-33bfda73785d>

8.2.References

- [1] wikipedia., "Online food ordering," *Online_food_ordering*, 2023,june 10.
- [2] S. P. Abhishek Singh, "Online Food Ordering System," *International Journal of Computer Applications*, vol. 180, 2017,december.
- [3] J. &. R. A. &. p. &. Riddy, "ONLINE FOOD ORDERING SYSTEM.," *10.13140/RG.2.2.25286.45121.*, 2022.
- [4] T. a. S. P. Deepa, "Online Food Ordering System," *nternational Journal of Emerging Technologies and Innovative Research*, p. 12, December 5, 2018.