`

EAT AND ROLL - AN ONLINE FOOD ORDERING SYSTEM

PRATHEESH J V

(71762251022)

SOORYA HARSHA P

(71762251035)

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

M.C.A (Master of Computer Applications) OF

ANNA UNIVERSITY

July 2023

DEPARTMENT OF COMPUTER APPLICATIONS COIMBATORE INSTITUTE OF TECHNOLOGY

(Autonomous Institution affiliated to Anna University)

COIMBATORE – 641014

`

COIMBATORE INSTITUTE OF TECHNOLOGY
(Autonomous Institution affiliated to Anna University)

COIMBATORE 641014

(Bonafide Certificate)

Mini-Project Work
Second Semester

EAT AND ROLL - AN ONLINE FOOD ORDERING SYSTEM

Bonfide record of Work done by

PRATHEESH J V

(71762251022)

SOORYA HARSHA P

(71762251035)

Submitted in partial fulfillment of the requirements
for the degree of
M.C.A. (Master of Computer Applications) of
Anna University

July 2023

_____ _____

Faculty Guide Head of the Department

Submitted for the viva-voce held on _____

_____ _____

Internal Examiner External Examiner

`

CONTENTS

CHAPTER	PAGE NO
ACKNOWLEDGEMENT	i
SYNOPSIS	ii
PREFACE	iii
I INTRODUCTION	
1.1 PROBLEM DEFINITION	
1.2 SYSTEM ENVIRONMENT	
II SYSTEM ANALYSIS	
2.1 SYSTEM DESCRIPTION	
2.2 LITERATURE STUDY	
2.3 USE CASE MODEL / BLOCK DIAGRAM (for IoT projects)	
2.4 SOFTWARE REQUIREMENTS SPECIFICATION	
III SYSTEM DESIGN	
3.1 ARCHITECTURAL DESIGN	
3.2 STRUCTURAL DESIGN	
3.3 BEHAVIOURAL DESIGN	
3.4 TABLE DESIGN	
3.5 USER INTERFACE DESIGN	
3.6 CODE DESIGN	
IV SYSTEM TESTING	
4.1 TEST CASES AND TEST REPORTS	
V SYSTEM IMPLEMENTATION VI CONCLUSION	
BIBLIOGRAPHY	
APPENDIX	
PUBLICATIONS	

`

ACKNOWLEDGEMENT

We take this opportunity with great pleasure, deep satisfaction and gratitude, to the contribution of many individuals for the successful completion of this mini project.

We express our hearty gratitude to **Dr.A. RAJESWARI**, Principal, Coimbatore Institute of Technology, Coimbatore.

We are extremely grateful to **Dr.R.S. SOMASUNDARAM**, Head, Department of Computer Applications, Coimbatore Institute of Technology,

We are thankful to our guide **Dr.J.B. JONA**, Assistant Professor, Department of Computer Applications, Coimbatore Institute of Technology, Coimbatore.

We also express our sincere thanks to all faculty and technical staff of the Department of Computer Applications for their motivation and encouragement.

`

ABSTRACT

- An Online Food Ordering System is proposed here which simplifies the food ordering process. The proposed system shows an user interface and update the menu with all available options so that it eases the customer work. Customer can choose multiple foods from multiple restaurants to make an order and can choose the payment method that they prefer .

- The order confirmation is sent to the customer. The order is placed in the queue and updated in the database and returned in real time. This system assists the staff to go through the orders in real time and process it efficiently with minimal errors.

- The online food ordering system will provide a number of benefits for both customers and restaurants. For customers, the system will make it easier and more convenient to order food online. For restaurants, the system will provide a way to manage orders more efficiently and track inventory more effectively.

`

PREFACE

I. INTRODUCTION

This chapter introduces the Online Food Ordering System. It elaborates the problem selection, problem statement, solution to the problem and the social impact.

II. TECHNICAL INFORMATION

This chapter contains information about the system requirements, software requirements specification and both functional and non-functional requirements.

III. SYSTEM DESIGN

This chapter contains details about the architecture of the Online Food Ordering System and the data flow diagram. It also contains sequence diagrams and all the tables that are used in this application.

IV. PROTOTYPE

This chapter explains the prototype UI design of the Online Food Ordering System and the code.

V. CONCLUSION

This chapter concludes by specifying the uses of the Online Food Ordering System if it is implemented successfully in real world.

`

I INTRODUCTION

1.1 PROBLEM DEFINITION

Inconvenience in the traditional food ordering process: The traditional process of ordering food through phone calls or in-person visits to restaurants can be time-consuming and inconvenient for customers. There is a need for an online platform that streamlines the ordering process, saving time and effort for customers.

Limited accessibility and availability: Many restaurants only accept orders during specific hours or have limited delivery options. This restricts customers from ordering their preferred meals at their convenience. The online food ordering system should provide a 24/7 accessible platform that allows customers to place orders and select delivery or pickup times according to their preferences.

Lack of transparency and information: Customers often face challenges in accessing comprehensive menus, detailed dish descriptions, ingredients, and pricing information. The system should provide a user-friendly interface that allows customers to browse through complete menus, view high-quality images of dishes, access nutritional information, and make informed decisions.

Inefficient order management for restaurants: Traditional methods of order management can lead to errors, miscommunication, and delays in processing orders. The online food ordering system should provide a streamlined order management interface for restaurants, facilitating accurate order processing, minimizing errors, and improving efficiency.

Limited payment options: Traditional food ordering methods often have limited payment options, such as cash on delivery. The system should integrate multiple secure online payment gateways to offer customers a variety of convenient payment methods, including credit/debit cards, digital wallets, and online banking.

`

1.2.1 HARDWARE REQUIREMENT SPECIFICATION

Technical descriptions of the computer components and capabilities are described by hardware specification. The hardware specification for the EatAnd Roll- An online food ordering system is given below.

- Processor: Intel i3
- Processor speed: 3.7 GHz
- RAM: 4 GB (Min)
- Hard Disk: 100 GB

1.2.2 SOFTWARE REQUIREMENTS SPECIFICATION

The software descriptions of the hackathon and its functionality are explained in the software specification. Software requirements are given below.

- Operating system : Windows OS
- Front-end : HTML, CSS,JavaScript, JQuery, Ajax,Python
- Back-end : PostgreSql
- Framework : django

`

II SYSTEM ANALYSIS

2.1 SYSTEM DESCRIPTION

EXISTING SYSTEM

The current system may have a limited number of restaurants available for ordering, restricting the choices for customers.There could be instances where the menu items or prices displayed on the platform do not match the actual offerings or prices at the restaurants, leading to confusion and dissatisfaction for customers.The existing system may have a complex or unintuitive user interface, making it difficult for customers to browse through menus, customize orders, or find relevant information.Customers may not receive timely updates on the status of their orders, causing frustration and uncertainty about when their food will be delivered.The system may experience glitches or errors during the payment process, leading to failed transactions or double charges, which can result in inconvenience and dissatisfaction for customers.The system may lack effective order management features, leading to delays or errors in processing orders, which can impact delivery times and overall customer experience.

PROPOSED SYSTEM

The proposed online food ordering system provides customers with the convenience of ordering food from their favourite restaurants through a web-based platform or mobile application. The system allows users to browse through a variety of menus, select dishes, customize their orders, and make payments electronically.

Upon accessing the online food ordering system, customers are required to create an account by providing their personal information, including name, contact details, and delivery address. Registered users can log in to the system using their credentials and access various functionalities. The system allows customers to search for nearby restaurants based on location, Users can view restaurant profiles, including their menu, prices. They can add items to their virtual shopping cart, customize their orders, and proceed to the checkout process. Customers are presented with various payment options, such as credit/debit card, digital wallets, or cash on delivery. Once the payment is completed, the order is confirmed, and the customer receives an order confirmation with details

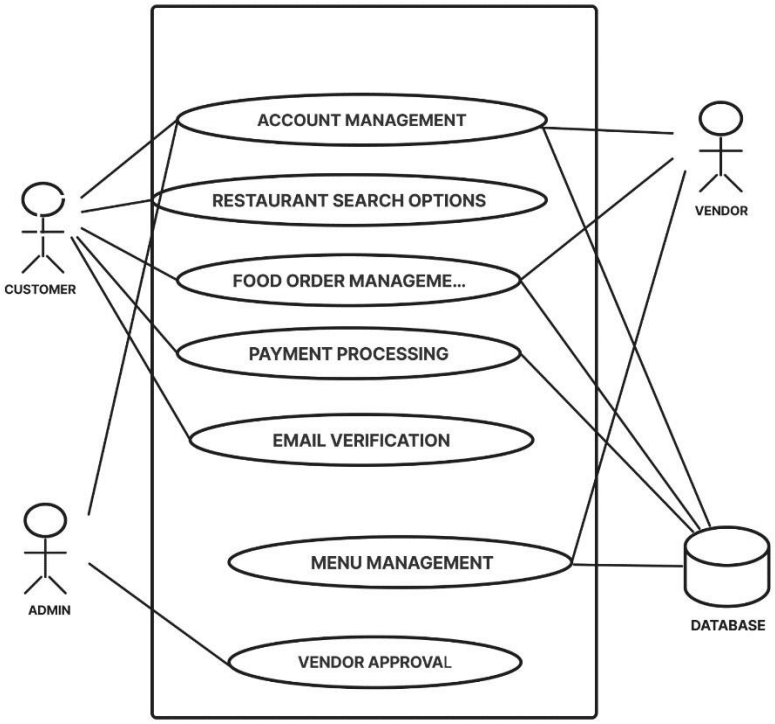
`

such as estimated delivery time. Overall the proposed online food ordering system simplifies the process of ordering food by providing a user-friendly platform, a wide selection of restaurants and menus, secure payment options.

2.2 USE CASE MODEL

The use case diagram is the simplest way of understanding the system. It summarizes the details of the system and the interactions with the system. The various components of the basic use case model are Actor, use cases, and associations. A use case means essential functionality of any working system. The following figure 2.1 depicts the use case diagram for the proposed system which provides a visual representation of the main functionalities and interactions Online Food Ordering System. It illustrates the different actors involved, such as donors and registered users, along with the system itself. The diagram showcases key use cases, including donors, registering the food details, requesting the food from donor, feedback from receiver. Each use case outlines the steps and interactions involved in completing specific tasks within the application. The use case diagram serves as a valuable tool for understanding the application's functionality, identifying user interactions, and providing a foundation for further development, testing, and documentation.

`



2.4 SOFTWARE REQUIREMENTS SPECIFICATION

FUNCTIONAL REQUIREMENT

- **User Registration and Authentication**: Users should be able to create accounts by providing their personal information, email account, and optional contact details. The system should verify user credentials during login to ensure secure access to their accounts**.**

- **Vendor Management**: Vendors should be able to register their profiles and provide necessary information such as contact details, menu items, and pricing. The system should facilitate the onboarding process for new vendors, including verification of their credentials. Vendors should be able to create, update, and manage their menus, including adding new dishes, setting prices, and specifying availability

`

- **Email Verification:** The user account will be activated only after the user has verified his account through the link sent to his/her email account. The vendor account verification status will also be updated through the email account.
- **Advanced Search:** The customers will be able to search for the vendors available in a particular range of area through the filter option implemented with the help of geocoding and places API. The exact latitude and longitude details can the fetched with the help of these API's. Similarly with the help of googles auto complete and places API's the expected places will be auto suggested in the searchbox.
- **Edit profile:** Both the customers and vendors will be able to edit their profiles including changing names,address,profile photo and location details etc.
- **Order Placement and Summary:** Users should be able to add items from multiple vendors to their shopping cart, specify quantities, and customize orders. Users should have a clear summary of their order, including items, quantities, and total cost, before proceeding to checkout.

NON- FUNCTIONAL REQUIREMENTS

- **Performance:** The system should provide quick responses to user actions, ensuring minimal delay in processing orders, payments, and other operations. The system should handle a large number of concurrent users and transactions without performance degradation**.**
- **Reliability:** The system should be highly available, minimizing downtime and ensuring continuous operation. The system should be able to recover gracefully from failures, ensuring that critical functions remain operational.Regular data backups should be performed to protect against data loss, and there should be a robust recovery mechanism in place.

- **Security:** Authentication and authorization: The system should have secure mechanisms for user authentication and authorization to ensure that only authorized personnel can

`

access sensitive information or perform critical actions. Sensitive user data, such as personal information and payment details, should be encrypted and stored securely to prevent unauthorized access.

- **Scalability:** The system should be able to handle increasing workload by adding more resources to individual components.The system should support adding more vendor partners and handling a growing number of users without significant performance degradation.
- **Usability:** The system should have an intuitive and user-friendly interface that allows vendors to easily manage their menus, track orders, and perform other relevant tasks. The system should provide clear documentation and training resources to help vendors understand and effectively use the system's features.

`

III SYSTEM DESIGN

3.1 ARCHITECTURL DESIGN

Architecture diagramming is the process of creating visual representations of software system components. In a software system, the term architecture refers to various functions, their implementations, and their interactions with each other. As software is inherently abstract, architecture diagrams visually illustrate the various data movements within the system. They also highlight how the software interacts with the environment around it.

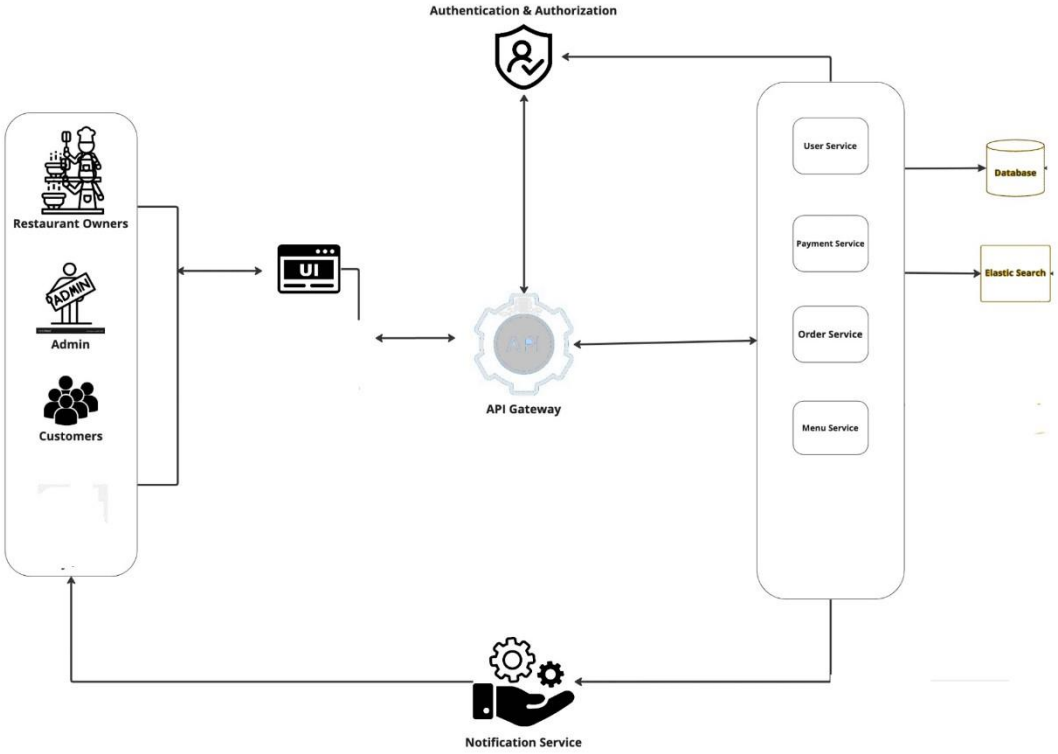


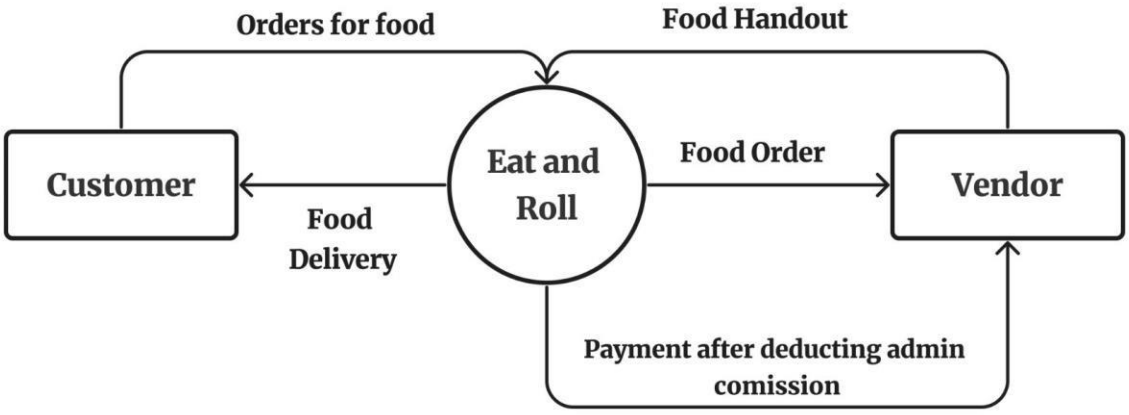
Fig.3.1 ARCHITECTURAL DESIGN

3.2 STRUCTURAL DESIGN

`

DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analysed or modelled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.

Fig.3.2 DFD LEVEL 0

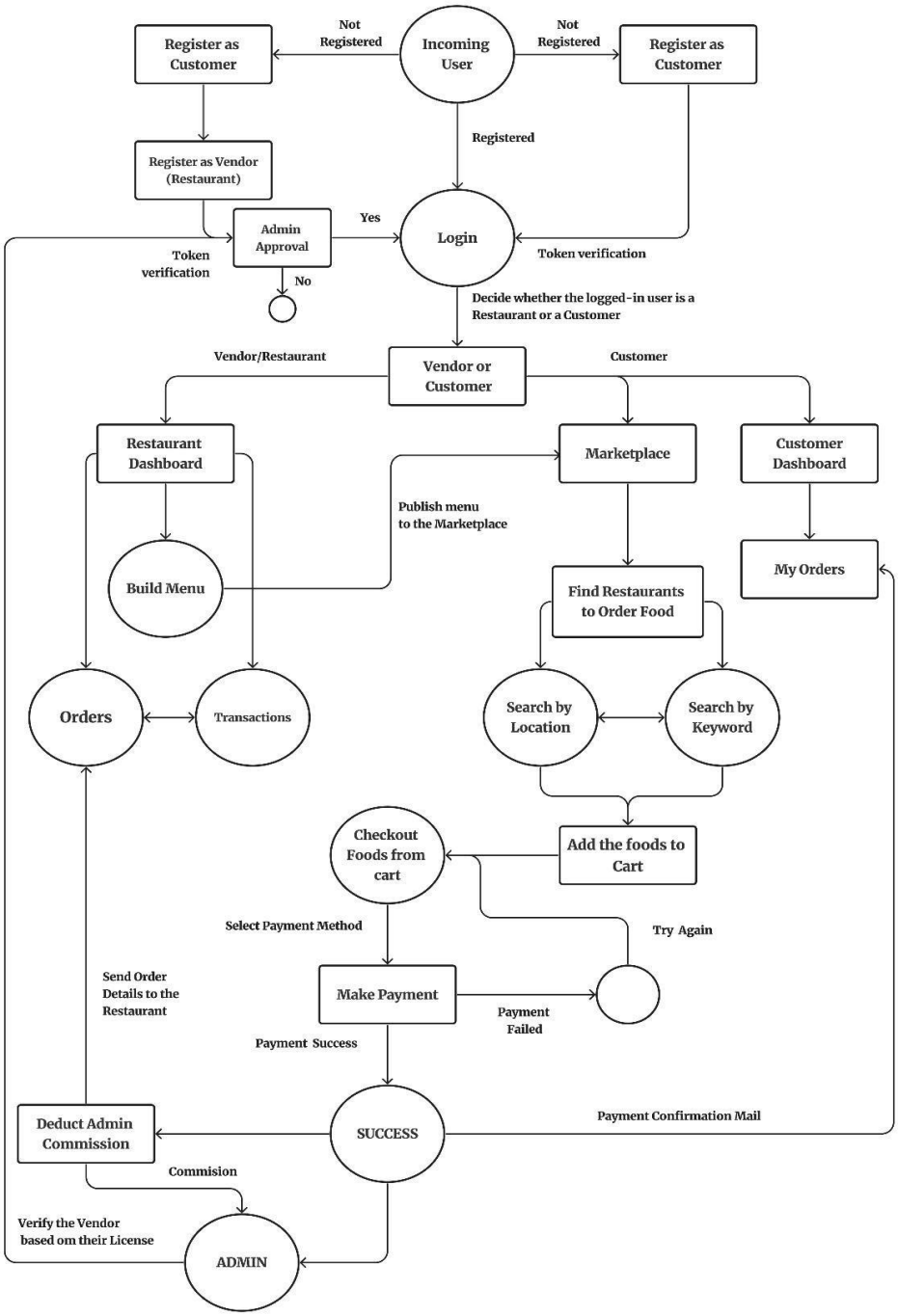


3.3 BEHAVIOURAL DESIGN

Activity Diagram

A activity diagram simply depicts visual representation of the flow of activities or processes within a system or project. It is a type of behavioral diagram in UML that shows the sequence and dependencies of various activities or actions. It represents the flow of activities or actions within a system or process. It visually depicts the sequential and parallel steps involved in completing a specific task or achieving a particular goal. Activity diagrams are commonly used in software development and business process modeling to analyze, design, and document workflows.

`



Activity Diagram Online Food Ordering

`

3.4 TABLE DESIGN

CUSTOMER TABLE

The customer table is used to keep track of all registered users on a website or application. This information can be used to identify between users, vendors, admin authenticate users, and send users notifications.

3.4.1 USER TABLE

Field Name	Data Type	Description	Constrains
User_id	Bigint	User ID	Primary key
First_name	Character varying (50)	First name of User	Not Null
Last_name	Character varying (50)	Last name of User	Not Null
Email	Character varying (100)	Email Id of the User	Unique = True
Username	Character varying (100)	Username for the account	Unique = True
Password	Character varying (128)	Password of the user	Not Null
Phone_number	Bigint	Phone number of the user	Not Null
Role	Smallint	1 represents Vendor account and 2 represents Customer account	Not Null
Is_admin	Bool	Tells whether the user is admin or not	Default = False
Is_active	Bool`	Tells whether the account is activated or not	Default = False

`

Created_date	Timestamp with timezone	Date and time of the account creation	Not Null
Modified_date	Timestamp with timezone	Date and time of the account modified	Not Null
Last_login	Timestamp with timezone	Date and time of the last login	Not Null

3.4.2 USER PROFILE TABLE

Field Name	**Data Type**	**Description**	**Constrain**
User_profile_id	Bigint	User profile ID	Primary key
User_id	Bigint	User ID	Foreign key
Profile_picture	Character varying (30)	URL of the user cover photo	Only jpg,jpeg and png
Cover_photo	Character varying (30)	URL of the user cover photo	Only jpg,jpeg and png
address	Character varying (30)	Address of the User	Not Null
Country	Character varying (30)	Country of the user	Not Null
State	Character varying (30)	State of the user	Not Null
City	Character varying (30)	City of the user	Not Null
Pin_code	int	Pin_code of the user address	Not Null
Created_at	Timestamp with time zone	Date and time of the account creation	Not Null
Modified_at	Timestamp with time zone	Date and time of the account modified	Not Null

`

3.4.3 VENDOR TABLE

Field Name	Data Type	Description	Constrain
Vendor_id	Bigint	Vendor ID	Primary Key
User_id	Bigint	User ID	Foreign key
User_profile_id	Bigint	User profile ID	Foreign key
Vendor_name	Character varying (50)	Name of the vendor	Not Null
Created_at	Timestamp with time zone	Date and time of the account creation	Not Null
Modified_at	Timestamp with time zone	Date and time of the account modified	Not Null

3.4.4 CATEGORY TABLE

Field Name	Data Type	Description	Constrain
Category_id	Bigint	Category ID	Primary Key
Vendor_id	Bigint	Vendor ID	Foreign Key
Category_name	Character varying (50)	Name of the food Category	Not Null
Description	Text	Description of the category	Null = True
Created_at	Timestamp with time zone	Date and time of the account creation	Not Null
Updated_at	Timestamp with time zone	Date and time of the category modified	Not Null

`

3.4.5 FOOD ITEM TABLE

Field Name	Data Type	Description	Constrain
Food_title_id	Bigint	Food ID	Primary Key
Category_id	Bigint	Category Id	Foreign Key
Vendor_id	Bigint	Vendor Id	Foreign Key
Food_title_slug	Character varying (50)	Slug of the Vendor	Unique = True
Vendor_license	Character varying (50)	URL of the vendor license	Not Null
Food_title	Character varying (50)	Name of the food	Not Null
Description	Text	Description of the Food	Null = True
Price	Numeric (10,2)	Price of the food item	Not Null
Created_at	Timestamp with time zone	Date and time of the Account created	Not Null
Updated_at	Timestamp with time zone	Date and time of the food item details last modified	Not Null

3.4.5 PAYMENT TABLE

Field Name	Data Type	Description	Constrain
Payment_id	Bigint	ID of the Payment	Primary Key
User_id	Bigint	ID of the user	Foreign Key
Transaction_id	Character varying(100)	ID of the transaction	Not Null

`

Payment_method	Character varying(100)	Method of Payment (Paypal / razorpay)	Not Null
Amount	Int	Amount of the ordered food	Not Null
Status	Character varying(100)	Success or Failure status of the payment	Not Null
Created_at	Timestamp with the time zone	Date and time of the payment	Not Null

3.4.6 ORDER TABLE

Field Name	**Data Type**	**Description**	**Constrain**
Order_id	Bigint	ID of the Order	Primary Key
Payment_id	Bigint	ID of the Payment	Foreign Key
User_id	Bigint	ID of the User	Foreign Key
Order_number	Character varying(20)	Order number	Unique = True
First_name	Character varying (50)	First name of User	Not Null
Last_name	Character varying (50)	Last name of User	Not Null
Email	Character varying (100)	Email Id of the User	Unique = True
Address	Character varying (30)	Address of the Customer	Not Null
Country	Character varying (30)	Country of the Customer	Not Null
State	Character varying (30)	State of the Customer	Not Null
Pin_code	Int	Pin_code of the Customer address	Not Null
Tax_data	Jasonb	GST and CSGT of the ordered food	Not Null
Total_tax	Int	Sum of both GST and CGST	Not Null
Payment_method	Character varying(20)	Method of payment (Paypal / Razorpay)	Not Null

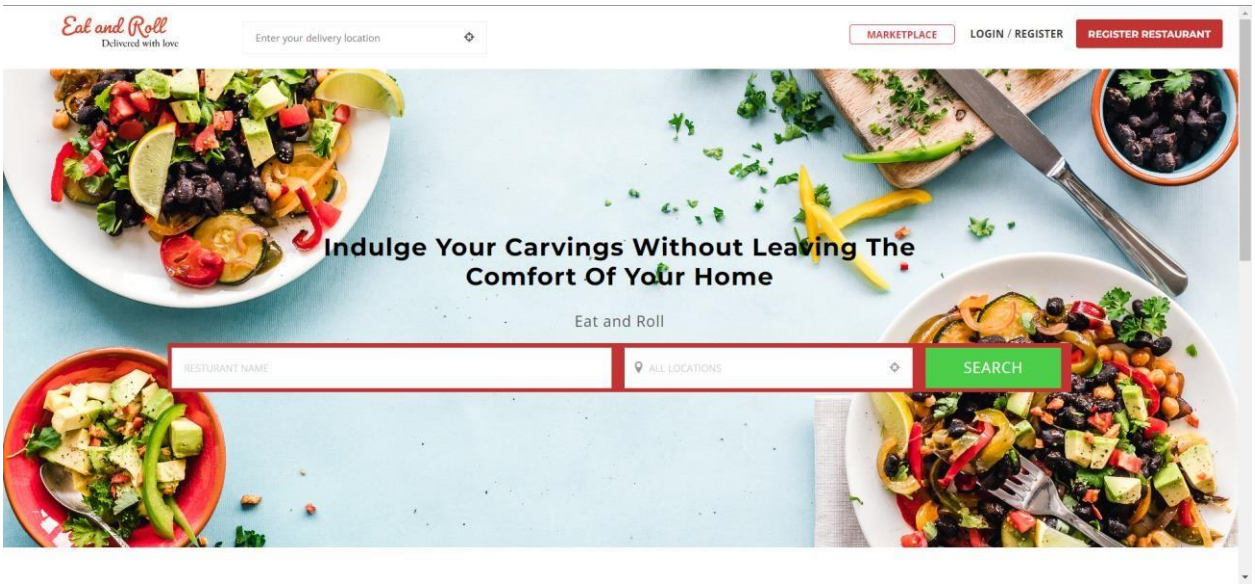
`

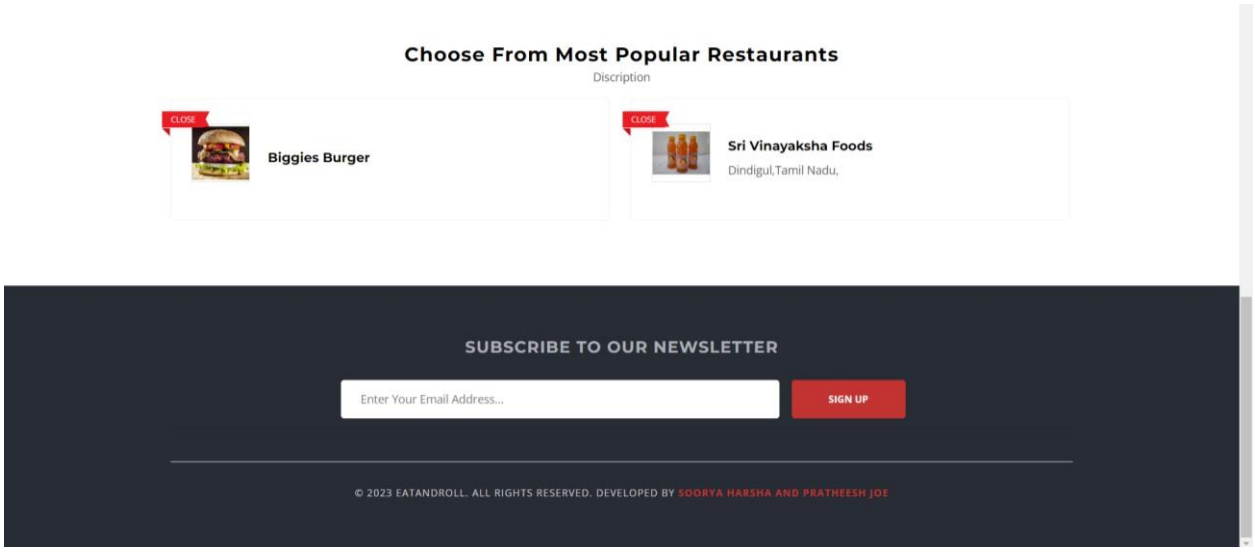
Is_ordered	Bool	Wheather the food is Ordered or not	Not Null
Created_at	Timestamp with time zone	Date and time of the Account created	Not Null
Updated_at	Timestamp with time zone	Date and time of the food item details last modified	Not Null

`

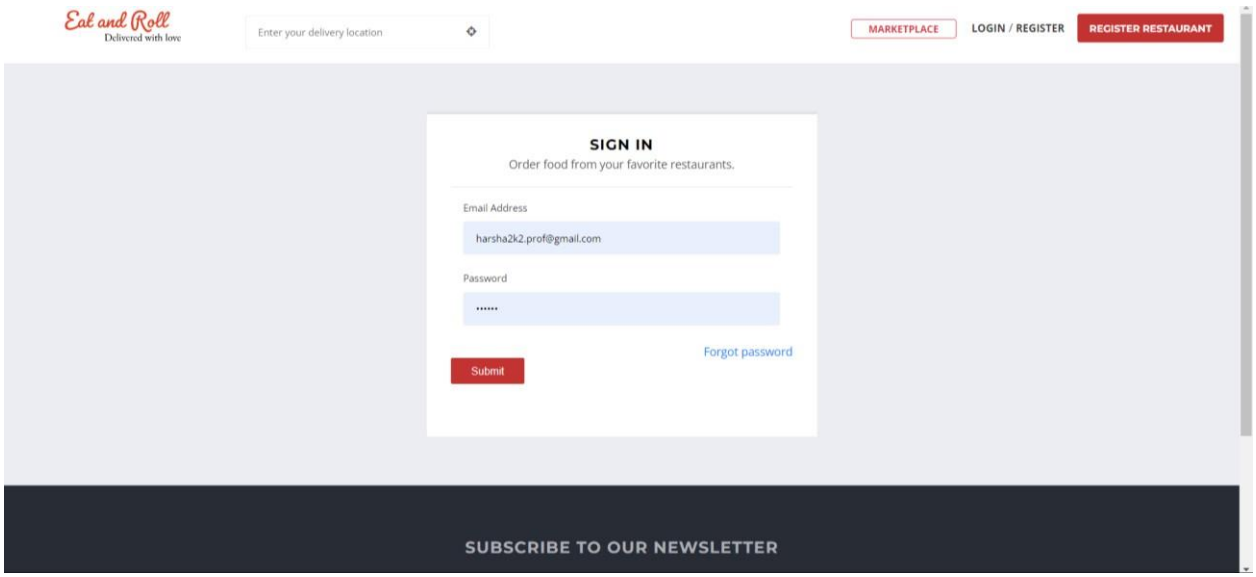
3.5 USER INTERFACE DESIGN :

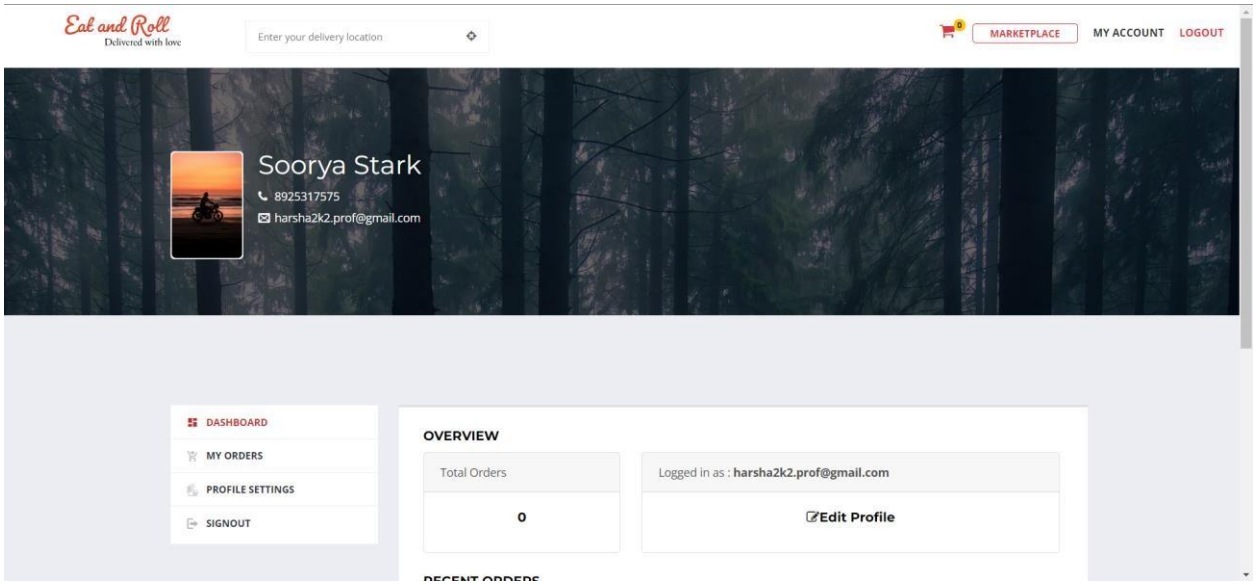
SCREEN SHOTS :



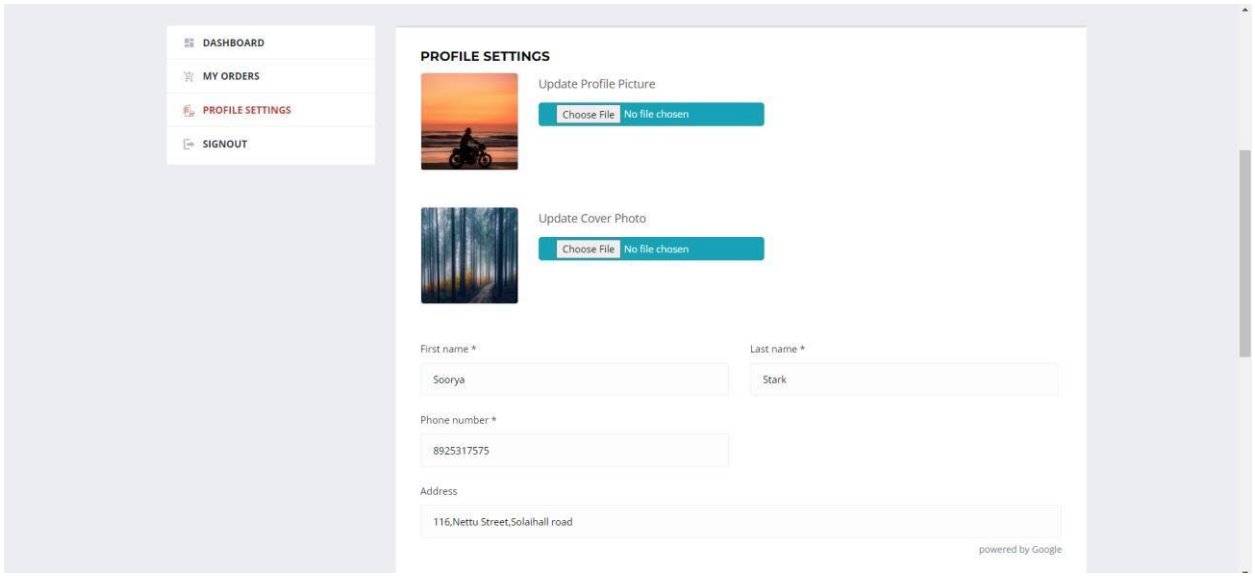


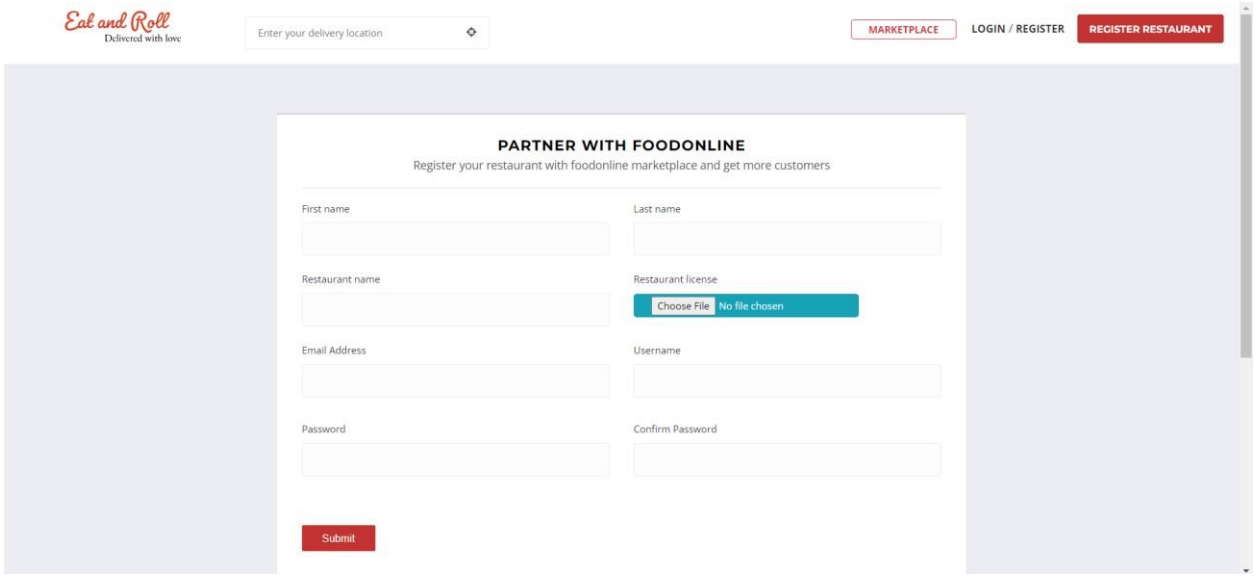
`


Eat and Roll
Delivered with love
Enter your delivery location
MARKETPLACE
LOGIN / REGISTER
REGISTER RESTAURANT
SIGN IN
Order food from your favorite restaurants.
Email Address
harsha2k2.prof@gmail.com
Password
Forgot password
Submit
SUBSCRIBE TO OUR NEWSLETTER

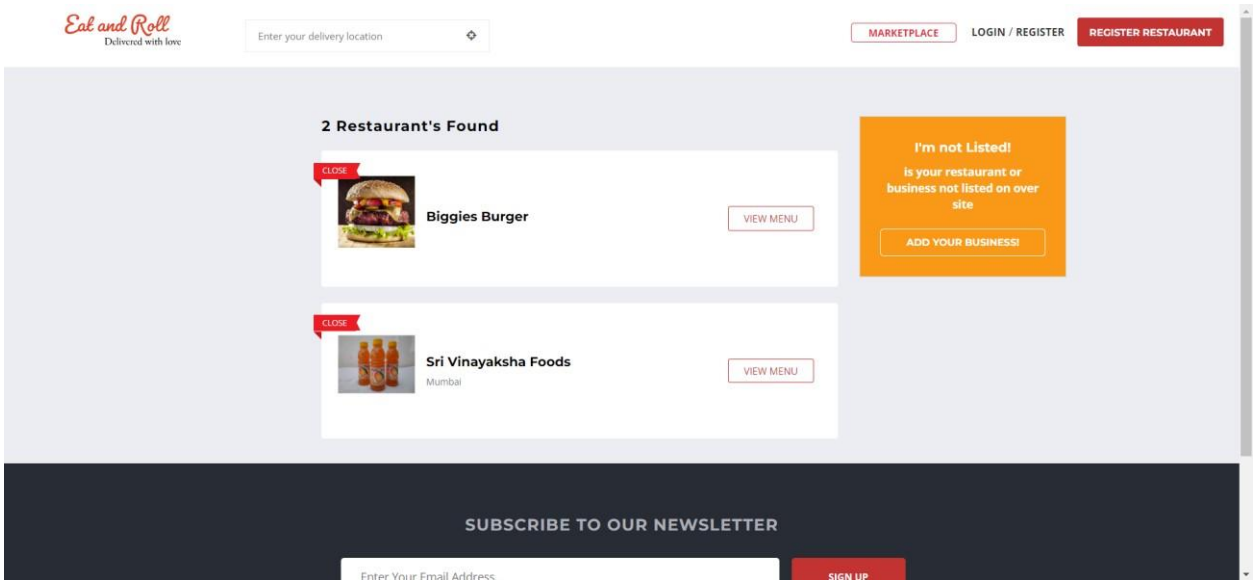

Eat and Roll
Delivered with love
Enter your delivery location
MARKETPLACE
MY ACCOUNT
LOGOUT
Soorya Stark
8925317575
harsha2k2.prof@gmail.com
DASHBOARD
MY ORDERS
PROFILE SETTINGS
SIGNOUT
OVERVIEW
Total Orders
0
Logged in as : harsha2k2.prof@gmail.com
Edit Profile

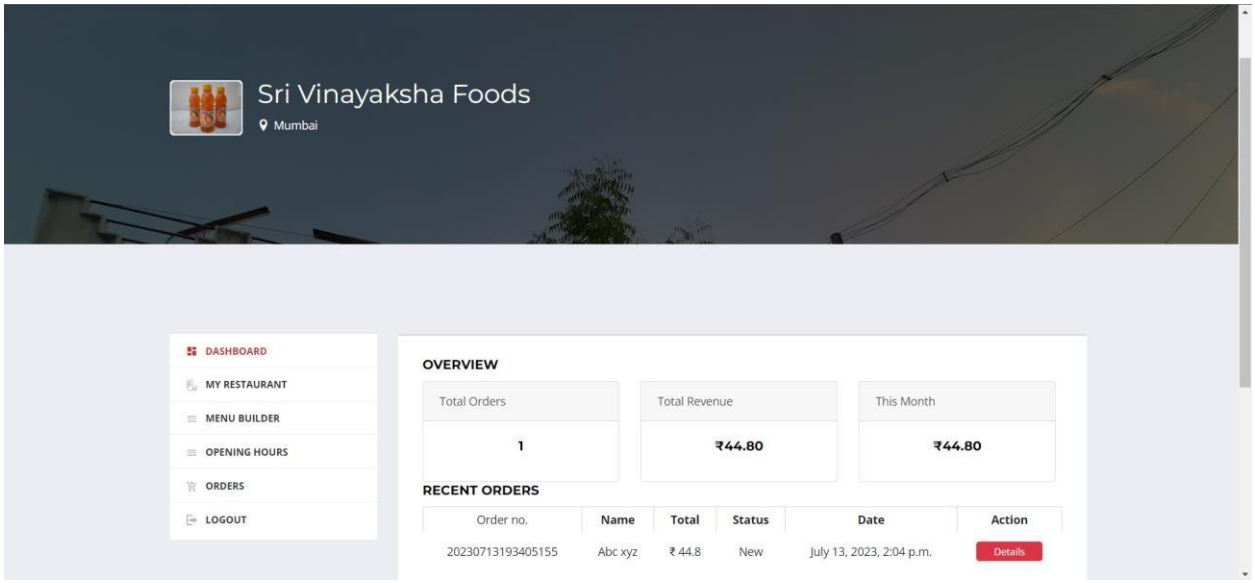
`


DASHBOARD
MY ORDERS
PROFILE SETTINGS
SIGNOUT
PROFILE SETTINGS
Update Profile Picture
Choose File
No file chosen
Update Cover Photo
Choose File
No file chosen
First name *
Soorya
Last name *
Stark
Phone number *
8925317575
Address
116,Nettu Street,Solaihall road
powered by Google

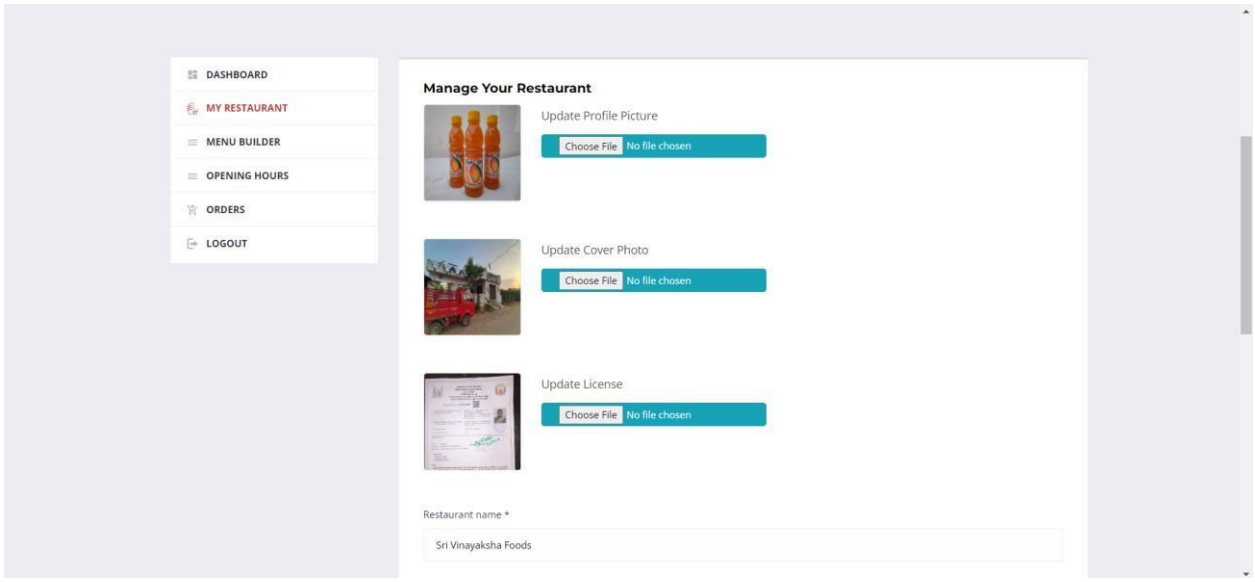

Eat and Roll
Delivered with love
Enter your delivery location
MARKETPLACE
LOGIN / REGISTER
REGISTER RESTAURANT
PARTNER WITH FOODONLINE
Register your restaurant with foodonline marketplace and get more customers
First name
Last name
Restaurant name
Restaurant license
Choose File
No file chosen
Email Address
Username
Password
Confirm Password
Submit

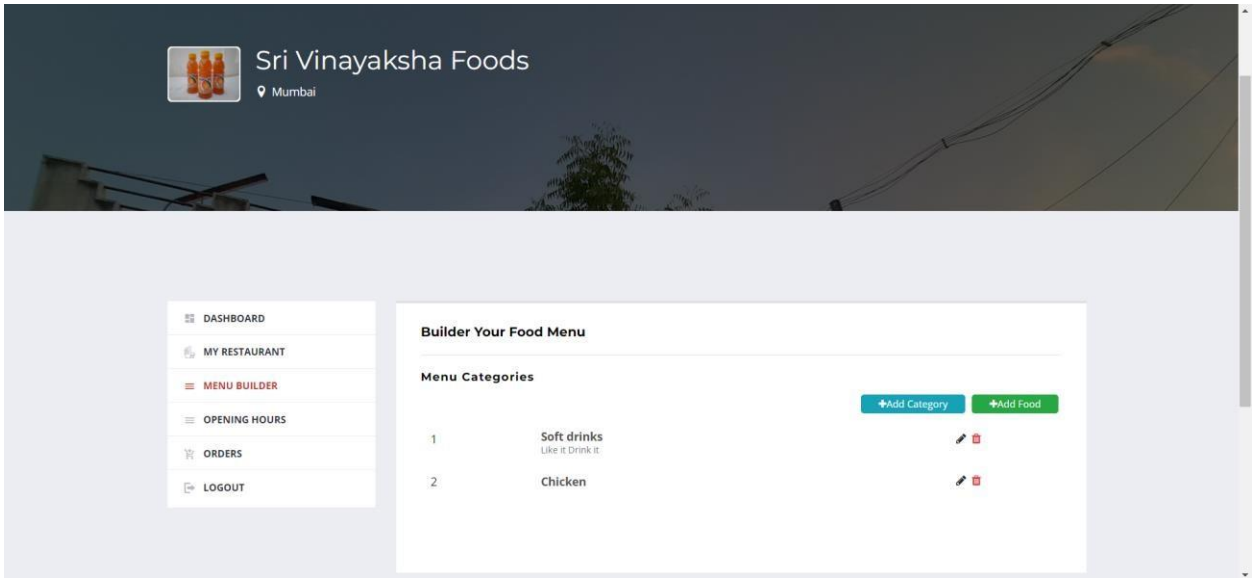
`


Eat and Roll
Delivered with love
Enter your delivery location
MARKETPLACE
LOGIN / REGISTER
REGISTER RESTAURANT
2 Restaurant's Found
CLOSE
Biggies Burger
VIEW MENU
I'm not Listed!
is your restaurant or business not listed on over site
ADD YOUR BUSINESS!
CLOSE
Sri Vinayaksha Foods
Mumbai
VIEW MENU
SUBSCRIBE TO OUR NEWSLETTER
Enter Your Email Address
SIGN UP

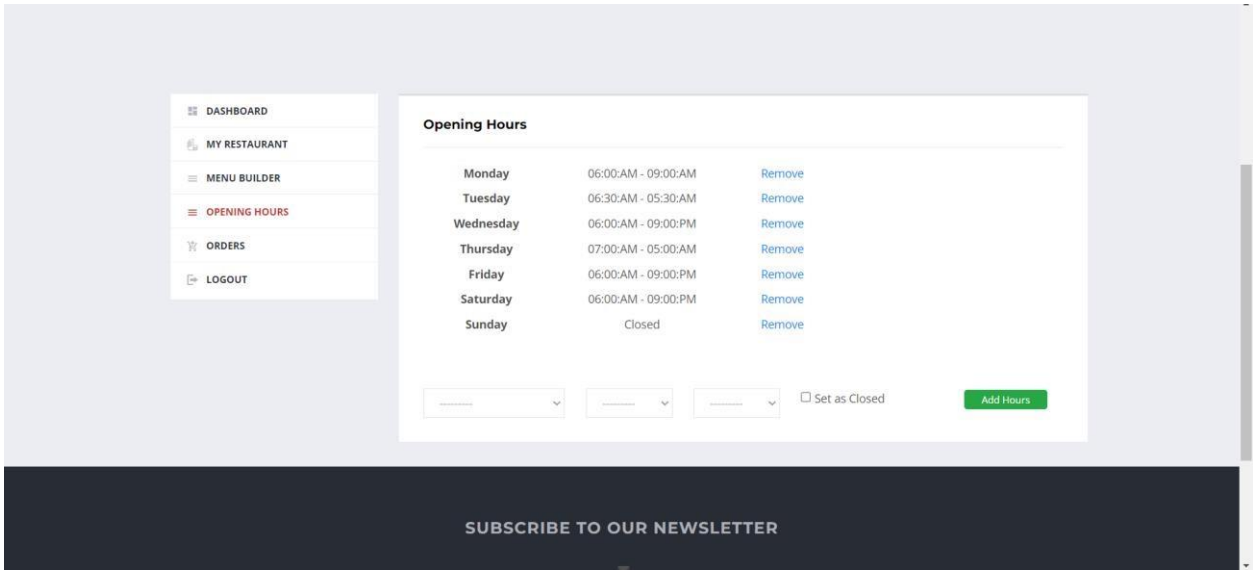

Sri Vinayaksha Foods
Mumbai
DASHBOARD
MY RESTAURANT
MENU BUILDER
OPENING HOURS
ORDERS
LOGOUT
OVERVIEW
Total Orders
1
Total Revenue
₹44.80
This Month
₹44.80
RECENT ORDERS
Order no.
Name
Total
Status
Date
Action
20230713193405155
Abc xyz
₹ 44.8
New
July 13, 2023, 2:04 p.m.
Details

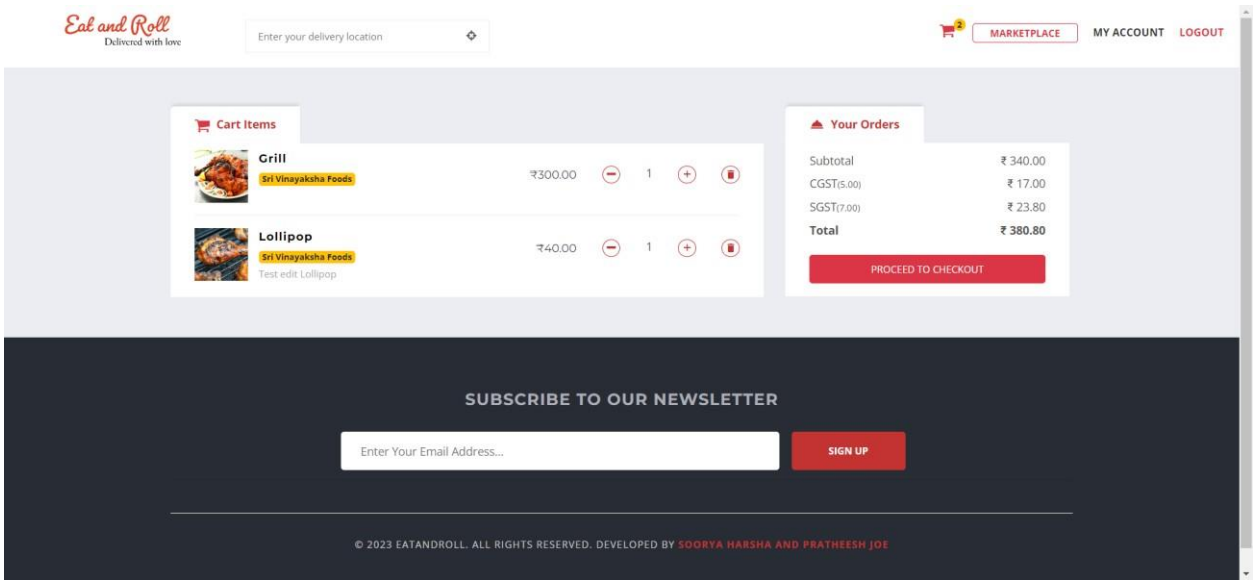
`


DASHBOARD
MY RESTAURANT
MENU BUILDER
OPENING HOURS
ORDERS
LOGOUT
Manage Your Restaurant
Update Profile Picture
Choose File No file chosen
Update Cover Photo
Choose File No file chosen
Update License
Choose File No file chosen
Restaurant name *
Sri Vinayaksha Foods

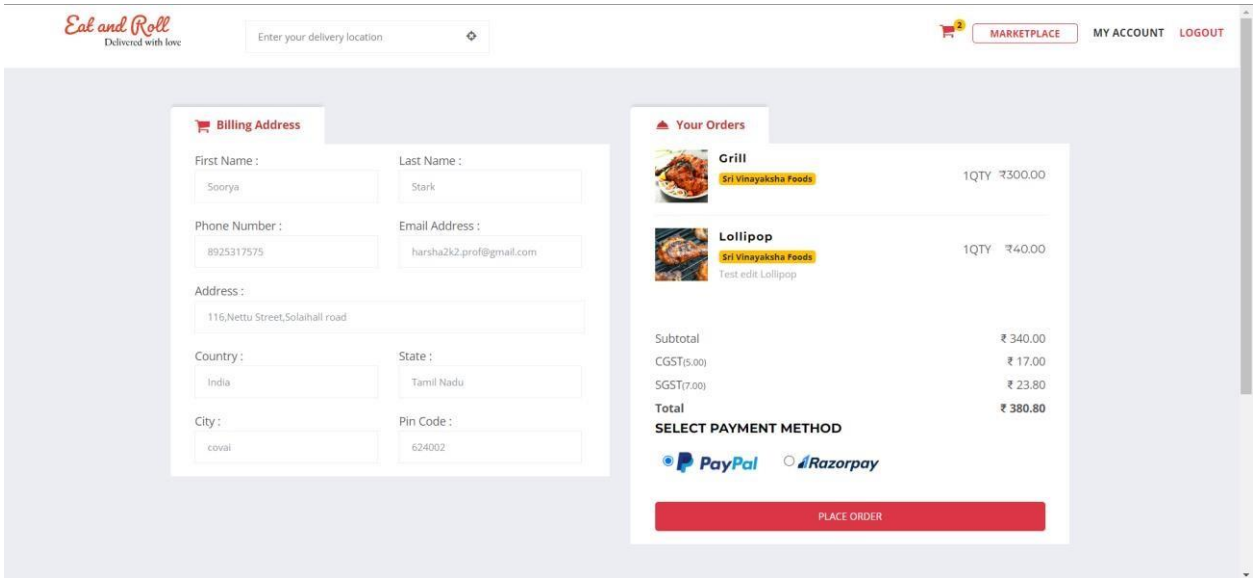

Sri Vinayaksha Foods
Mumbai
DASHBOARD
MY RESTAURANT
MENU BUILDER
OPENING HOURS
ORDERS
LOGOUT
Builder Your Food Menu
Menu Categories
+Add Category
+Add Food
1
Soft drinks
Like it Drink it
2
Chicken

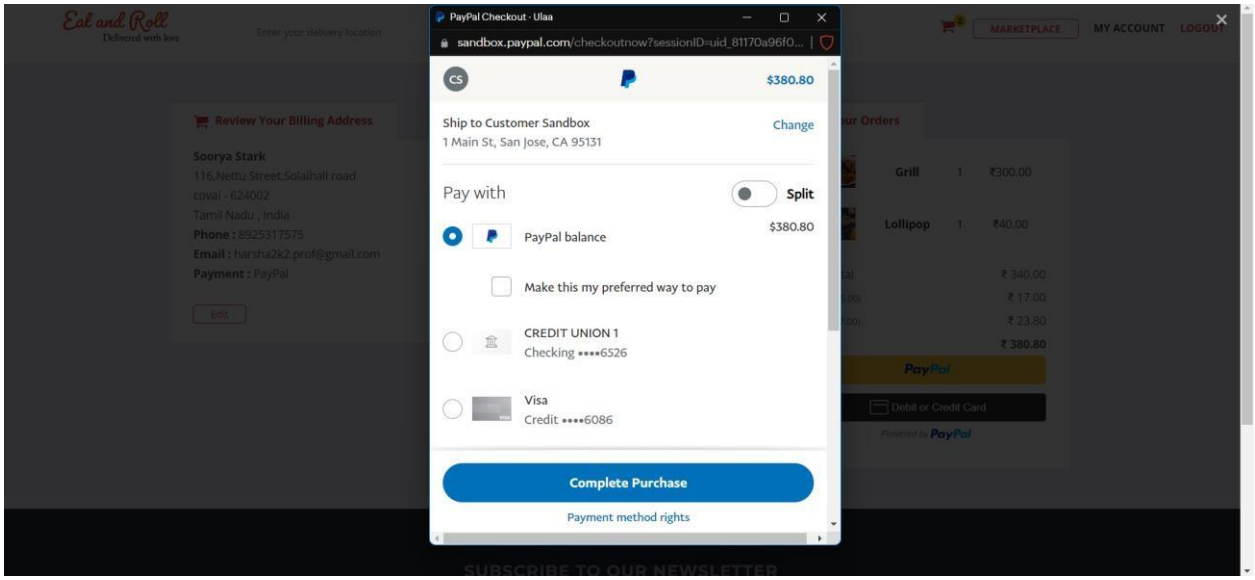
`


DASHBOARD
MY RESTAURANT
MENU BUILDER
OPENING HOURS
ORDERS
LOGOUT
Opening Hours
Monday 06:00:AM - 09:00:AM Remove
Tuesday 06:30:AM - 05:30:AM Remove
Wednesday 06:00:AM - 09:00:PM Remove
Thursday 07:00:AM - 05:00:AM Remove
Friday 06:00:AM - 09:00:PM Remove
Saturday 06:00:AM - 09:00:PM Remove
Sunday Closed Remove
Set as Closed
Add Hours
SUBSCRIBE TO OUR NEWSLETTER

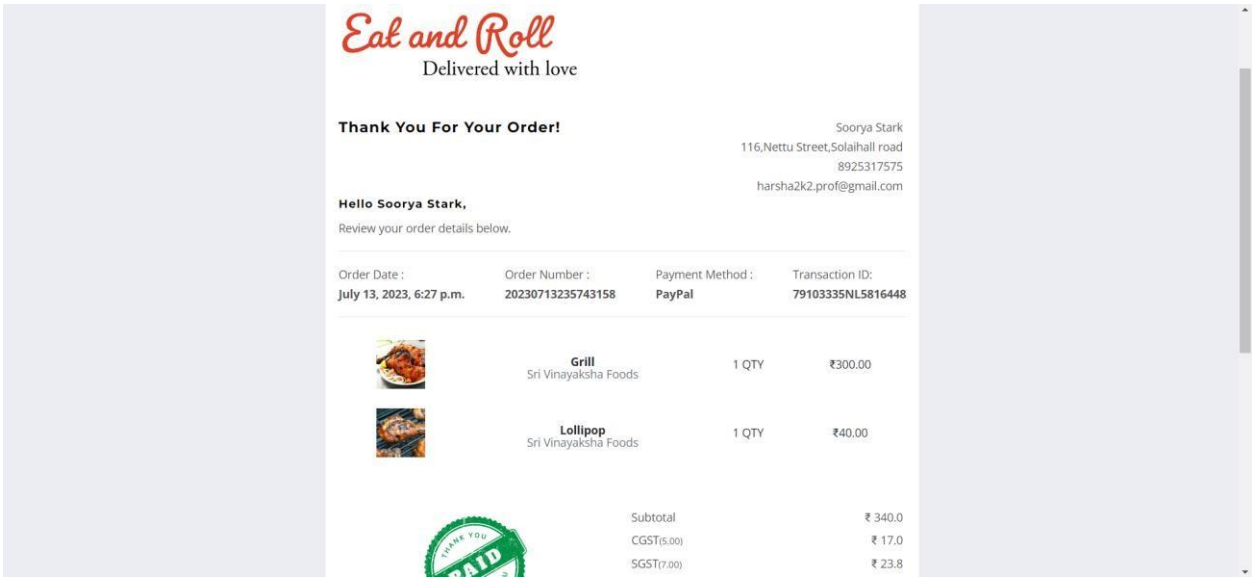

Eat and Roll
Delivered with love
Enter your delivery location
MARKETPLACE
MY ACCOUNT
LOGOUT
Cart Items
Grill
Sri Vinayaksha Foods
₹300.00
1
Lollipop
Sri Vinayaksha Foods
Test edit Lollipop
₹40.00
1
Your Orders
Subtotal ₹ 340.00
CGST(5.00) ₹ 17.00
SGST(7.00) ₹ 23.80
Total ₹ 380.80
PROCEED TO CHECKOUT
SUBSCRIBE TO OUR NEWSLETTER
Enter Your Email Address...
SIGN UP
© 2023 EATANDROLL. ALL RIGHTS RESERVED. DEVELOPED BY SOORYA HARSHA AND PRATHEESH JOE

`


Eat and Roll
Delivered with love
Enter your delivery location
MARKETPLACE
MY ACCOUNT
LOGOUT
Billing Address
First Name :
Soorya
Last Name :
Stark
Phone Number :
8925317575
Email Address :
harsha2k2.prof@gmail.com
Address :
116,Nettu Street,Solaihall road
Country :
India
State :
Tamil Nadu
City :
covai
Pin Code :
624002
Your Orders
Grill
Sri Vinayaksha Foods
1QTY ₹300.00
Lollipop
Sri Vinayaksha Foods
Test edit Lollipop
1QTY ₹40.00
Subtotal ₹ 340.00
CGST(5.00) ₹ 17.00
SGST(7.00) ₹ 23.80
Total ₹ 380.80
SELECT PAYMENT METHOD
PayPal
Razorpay
PLACE ORDER


Eat and Roll
Delivered with love
Enter your delivery location
MARKETPLACE
MY ACCOUNT
LOGOUT
Review Your Billing Address
Soorya Stark
116,Nettu Street,Solaihall road
covai - 624002
Tamil Nadu , India
Phone : 8925317575
Email : harsha2k2.prof@gmail.com
Payment : PayPal
Edit
Your Orders
Grill 1 ₹300.00
Lollipop 1 ₹40.00
₹ 340.00
₹ 17.00
₹ 23.80
₹ 380.80
PayPal
Debit or Credit Card
Powered by PayPal
SUBSCRIBE TO OUR NEWSLETTER
PayPal Checkout - Ulaa
sandbox.paypal.com/checkoutnow?sessionID=uid_81170a96f0...
CS
$380.80
Ship to Customer Sandbox
1 Main St, San Jose, CA 95131
Change
Pay with
Split
$380.80
PayPal balance
Make this my preferred way to pay
CREDIT UNION 1
Checking ••••6526
Visa
Credit ••••6086
Complete Purchase
Payment method rights

`



Eat and Roll
Delivered with love

Thank You For Your Order!

Soorya Stark
116,Nettu Street,Solaihall road
8925317575
harsha2k2.prof@gmail.com

Hello Soorya Stark,

Review your order details below.

Order Date :	Order Number :	Payment Method :	Transaction ID:
July 13, 2023, 6:27 p.m.	**20230713235743158**	**PayPal**	**79103335NL5816448**

Grill Sri Vinayaksha Foods		1 QTY	₹300.00
Lollipop Sri Vinayaksha Foods		1 QTY	₹40.00

Subtotal	₹ 340.0
CGST(5.00)	₹ 17.0
SGST(7.00)	₹ 23.8

Thank you for ordering with us

EM **EatAndRoll Marketplace <eatandroll23@gmail.com>**
23:58

To: harsha2k2.prof@gmail.com

Hi Soorya,

Thank you for your order.
Your order number : 20230713235743158
Transaction ID : 79103335NL5816448

`

3.6 CODE DESIGN :

USER AND VENDOR REGISTRATION :

```
def registerUser(request):     if request.user.is_authenticated:
messages.warning(request, "You are already logged in")
return redirect("custDashboard")     elif request.method ==
"POST":         print(request.POST)         form =
UserForm(request.POST)         if form.is_valid():
            first_name = form.cleaned_data["first_name"]
last_name = form.cleaned_data["last_name"]
username = form.cleaned_data["username"]
email = form.cleaned_data["email"]             password
= form.cleaned_data["password"]             user =
User.objects.create_user(
first_name=first_name,
last_name=last_name,                email=email,
username=username,                password=password,
            )
            user.role = User.CUSTOMER
            user.save()

            # send verification email

            mail_subject = "Please activate your account" email_template =
            "accounts/emails/account_verification_email.html"
            send_verification_email(request, user, mail_subject, email_template)


            messages.success(request, "Your account has been registered successfully !")
return redirect("registerUser")         else:
            print("Invalid form")
print(form.errors)     else:
```

```
`
      form = UserForm()
context = {
"form": form,
   }
   return render(request, "accounts/registerUser.html", context)


def registerVendor(request):     if
request.user.is_authenticated:
messages.warning(request, "You are already logged in")
return redirect("myAccount")

   elif request.method == "POST":
      # store the data         form = UserForm(request.POST)
v_form = VendorForm(request.POST, request.FILES)
if form.is_valid() and v_form.is_valid():
          first_name = form.cleaned_data["first_name"]
last_name = form.cleaned_data["last_name"]
username = form.cleaned_data["username"]
email = form.cleaned_data["email"] password =
form.cleaned_data["password"] user =
User.objects.create_user(
first_name=first_name,
last_name=last_name,                email=email,
username=username,                password=password,
          )
          user.role = User.VENDOR             user.save()
vendor = v_form.save(commit=False)             vendor.user =
user             vendor_name =
v_form.cleaned_data["vendor_name"]

          vendor.vendor_slug = slugify(vendor_name) + "-" + str(user.id)
```

`

```
        user_profile = userProfile.objects.get(user=user)
vendor.user_profile = user_profile          vendor.save()

        # send verification email          mail_subject = "Please activate your
account"          email_template =
"accounts/emails/account_verification_email.html"
send_verification_email(request, user, mail_subject, email_template)

        messages.success(
request,
            "You account has been registered successfully ! Please wait for the approval.",
        )
        return        redirect("registerVendor")
else:
        print("invalid form")
        print(form.errors)    else:
    form      =      UserForm()
v_form = VendorForm()

  context      =      {
"form": form,
    "v_form": v_form,
  }
  return render(request, "accounts/registerVendor.html", context)
```

ORDER PLACEMENT :

```
@login_required(login_url="login") def
place_order(request):
  cart_items = Cart.objects.filter(user=request.user).order_by("created_at")
cart_count = cart_items.count()    if cart_count <= 0:
    return redirect("marketplace")
```

`

```
   vendors_ids    =    []
for i in cart_items:
      if i.fooditem.vendor.id not in vendors_ids:
vendors_ids.append(i.fooditem.vendor.id)
print(vendors_ids)

   get_tax = Tax.objects.filter(is_active=True)
subtotal = 0     total_data = {}     k = {}     for
i in cart_items:
      fooditem    =    FoodItem.objects.get(pk=i.fooditem.id,    vendor_id__in=vendors_ids)
v_id = fooditem.vendor.id
      if v_id in k:
         subtotal = k[v_id]           subtotal +=
fooditem.price * i.quantity           k[v_id] =
subtotal        else:
         subtotal    =    fooditem.price    *    i.quantity
k[v_id] = subtotal

      # calculate tax_data
tax_dict = {}        for i
in get_tax:
         tax_type = i.tax_type           tax_percentage = i.tax_percentage
tax_amount = round((tax_percentage * subtotal) / 100, 2)
tax_dict.update({tax_type: {str(tax_percentage): str(tax_amount)}})       #
print(tax_dict)

      # Construct the total data        total_data.update({fooditem.vendor.id:
{str(subtotal): str(tax_dict)}})     print(total_data)

   subtotal = get_cart_amounts(request)["subtotal"]
total_tax = get_cart_amounts(request)["tax"]     grand_total
```

```
= get_cart_amounts(request)["grand_total"]     tax_data =
get_cart_amounts(request)["tax_dict"]

   if request.method == "POST":
      form = OrderForm(request.POST)         if
form.is_valid():             order = Order()
order.first_name = form.cleaned_data["first_name"]
order.last_name = form.cleaned_data["last_name"]
order.phone = form.cleaned_data["phone"]
order.email = form.cleaned_data["email"]
order.address = form.cleaned_data["address"]
order.country = form.cleaned_data["country"]
order.state = form.cleaned_data["state"]             order.city =
form.cleaned_data["city"]             order.pin_code =
form.cleaned_data["pin_code"]             order.user =
request.user             order.total = grand_total
order.tax_data = json.dumps(tax_data)
order.total_data = json.dumps(total_data)
order.total_tax = total_tax             order.payment_method =
request.POST["payment_method"]             order.save()
order.order_number = generate_order_number(order.id)
order.vendors.add(*vendors_ids)             order.save()

        # Razorpay payment

        DATA = {
            "amount": float(order.total) * 100,
            "currency": "INR",
            "receipt": "receipt #" + order.order_number,
            "notes": {"key1": "value3", "key2": "value2"},
        }
```

```
`
        rzp_order        =        client.order.create(data=DATA)
rzp_order_id = rzp_order["id"]

        context      =      {
"order": order,
            "cart_items": cart_items,
            "rzp_order_id": rzp_order_id,
            "RZP_KEY_ID": RZP_KEY_ID,
            "rzp_amount": float(order.total) * 100,
        }

        return render(request, "order/place_order.html", context)

else:
        print(form.errors)

   return render(request, "order/place_order.html")
```

PAYMENT PROCESSING :

```
@login_required(login_url="login") def
payments(request):
   # check if the request is ajax
    if
(
      request.headers.get("x-requested-with")    ==    "XMLHttpRequest"
and request.method == "POST"
   ):
      # store the payment details in the payment model
order_number = request.POST.get("order_number")
transaction_id = request.POST.get("transaction_id")
payment_method = request.POST.get("payment_method")
status = request.POST.get("status")
```

```
`
    print(order_number, transaction_id, payment_method, status)    order =
Order.objects.get(user=request.user,order_number=order_number)    print(order)    payment = Payment(
user=request.user,    transaction_id=transaction_id,    payment_method=payment_method,
amount=order.total,    status=status,
    )
    payment.save()

    # update the Oder model is_order to tru
order.payment = payment
order.is_ordered = True    order.save()

    # move the cart items to order food model
cart_items = Cart.objects.filter(user=request.user)
for item in cart_items:
        ordered_food = OrderedFood()
ordered_food.order = order    ordered_food.payment
= payment    ordered_food.user = request.user
ordered_food.fooditem = item.fooditem
ordered_food.quantity = item.quantity
ordered_food.price = item.fooditem.price
        ordered_food.amount = item.fooditem.price * item.quantity    # total amount
ordered_food.save()

    # send order confirmation email to customer
    mail_subject = "Thank you for ordering with us"
mail_template = "order/order_confirmation_email.html"
context = {    "user": request.user,
        "order": order,
        "to_email": order.email,
    }
    send_notification(mail_subject, mail_template, context)
```

```
`
        # send order received email to vendor
mail_subject = "You have received a new order"
mail_template = "order/new_order_received.html"
to_emails = []          for i in cart_items:                if
i.fooditem.vendor.user.email not in to_emails:
to_emails.append(i.fooditem.vendor.user.email)

        context        =        {
"order": order,
            "to_email": to_emails,
        }
        send_notification(mail_subject, mail_template, context)

        # clear the cart if the payment is success
cart_items.delete()         response = {
            "order_number": order_number,
            "transaction_id": transaction_id,
        }
        return JsonResponse(response)

        # Return back to ajax with the status success or failure

    return HttpResponse("Payments view")
```

AFTER ORDER CONFIRMATIONS:

```
def order_complete(request):
    order_number = request.GET.get("order_no")
transaction_id = request.GET.get("trans_id")      try:
        order = Order.objects.get(
order_number=order_number,
payment__transaction_id=transaction_id,
is_ordered=True,
```

```
        )
        ordered_food = OrderedFood.objects.filter(order=order)
subtotal = 0          for item in ordered_food:
            subtotal += item.price * item.quantity

        tax_data = json.loads(order.tax_data)
context = {              "order": order,
            "ordered_food": ordered_food,
            "subtotal": subtotal,
            "tax_data": tax_data,
        }
        return render(request, "order/order_complete.html", context)

    except:
        return redirect("home")
```

IV SYSTEM TESTING

`

4.1 Unit Testing

Unit testing focuses verification efforts on the smallest unit of software design, the module. This is also known as "Module Testing" The modules are tested separately this testing is carried out during programming stage itself. In this step each module is found to be working satisfaction as regard to the expected output from the module.

4.2 Integration Testing

Integration testing focuses on the design and construction of the software architecture. Data can be lost across an interface; one module can have adverse effect on another sub functions and show on. Thus, integration testing is a systematic technique for constructing test to uncover errors associated with in the interface. In this project, all the modules are companied and then the entire program is tested as a whole.

4.3 User Acceptance Testing

User acceptance testing of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keep in touch with the prospective system user at time of developing and making changes wherever required.

4.4 Validation Testing

Validation testing is the requirement established as a part of software requirement analysis is validated against the software that has been constructed. This test provides the final assurance whether the software needs all functional, behavioural and performance requirements Thus, the proposed system under consideration has been tested by using validation testing and found to be working satisfactory.

`

VI CONCLUSION

FUTURE SCOPE

The future enhancement of the online food ordering project aims to elevate the user experience, expand the service offerings, and embrace emerging technologies. The following are the key areas of focus for future development:

- Advanced Personalization: Implement advanced personalization techniques using AI and machine learning algorithms to analyze user preferences, order history, and behavior. This will enable the platform to provide personalized recommendations, customized promotions, and tailored menus based on individual preferences, dietary restrictions, and previous orders.
- Voice and Natural Language Processing: Integrate voice recognition and natural language processing capabilities to enable customers to place orders, make inquiries, and interact with the system using voice commands. This enhancement will offer a hands-free and intuitive ordering experience through smart devices and voice assistants.
- Social Media Integration: Enable seamless integration with popular social media platforms to allow customers to share their food experiences, write reviews, and recommend restaurants to their friends. This feature will enhance user engagement, promote user-generated content, and attract new customers through social media referrals.
- Live Order Tracking: Implementing a live order tracking feature allows customers to track the progress of their delivery in real-time. This can be achieved by integrating GPS technology with the delivery personnel's mobile devices. Customers can access a map view that shows the current location of the delivery person and their estimated time of arrival.

`

BIBLIOGRAPHY

- Garg, R., & Bajaj, S. (2020). A Comprehensive Study of Online Food Ordering Systems. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 9(1), 2385-2389.
- Chai, J., Qu, H., & Zhong, L. (2019). Understanding Online Food Delivery Apps: An Integrated Model of Technology Acceptance and Service Quality. International Journal of Hospitality Management, 77, 401-413.
- Singh, A., & Aggarwal, P. (2017). Online Food Ordering: A Review and Future Directions. Journal of Indian Business Research, 9(1), 30-59.