Online Food ordering System

BTech/III Year CSE/V Semester

15CSE302/Database Management Systems

Project Final Review

Rollno	Name
CB.EN.U4CSE18415	C.Dhanush
CB.EN.U4CSE18439	V.V.Mithun Rosinth

Amrita School of Engineering, Coimbatore

Department of Computer Science and Engineering

2020 -2021 Odd Semester

Table of Contents

Abstract

There is a lot of scope for online food ordering business and we can tap it to the max extent possible as everyone has access to an online ordering facility via the internet. Food business usually will have high demand and hence online business prospect for food ordering should be profitable. The purpose of this project is to provide an easily accessible interface wherein the customer can view and place the order easily.

The customer can register initially with minimum details and will be allowed to check the menu items before ordering them, adding them to cart and submit the order. The system records the details in the database so that it will be easy to retrieve later. The users of the system also include employee/admin who will handle info related to product addition and assigning vehicle for placed orders.

The users of the system include the customers and the employees. The employees of the system are responsible for updating the menu items as well as the delivery of the item to a particular address. The customers will visit the website, check for the items available in the menu, order for one or more items in the menu. All the activities such as ordering items online, delivery of the items by employees, the vehicle used to deliver the items etc. will be recorded in the database for all the events.

Business Rules

Flow:

USER:

- The user enters the registered phone no and an OTP is pushed to the phone
- Then the user can login by entering the received OTP
- If not a registered user, he/she can register by filling up address and phone.no
- After successful login the user is presented with the Menu
- The user picks the items and the quantity required and adds them to the cart
- From the Cart the user can proceed to place the order by selecting the desired payment method and editing the address if required
- Once the order is placed it is moved into the Orders sections
- The status of the order is updated once it is delivered to the user

Desk Agent:

- The Desk Agent login accepts Employee ID and the assigned password (Stored in the DB) as the credentials for the Desk Employee
- Once logged in, the employee can see the orders that have not yet been pushed into the delivery process on the dashboard
- The Desk agent can choose any one of the orders and choose a Delivery Agent for it to be assigned to.
- Once the ordered is assigned, the order is displayed on the dashboard of the respective Delivery agent
- The Desk Employee also has the access to edit the Menu to change Availability,Name,Price of existing items and add new items

Delivery Agent:

- The Order pushed from the Desk agent dashboard is displayed on the dashboard of the Delivery Agent
- The Delivery agent must click on the Delivered button on the dashboard once the order is delivered
- Now this delivery agent is also again available to take up orders

Preview of the Project:

There is a lot of scope for online food ordering business and we can tap it to the max extent possible as everyone has access to an online ordering facility via the internet. Food business usually will have high demand and hence online business prospect for food ordering should be profitable. The purpose of this project is to provide an easily accessible interface wherein the customer can view and place the order easily.

Technologies Used:

Frontend: NodeJS, Material UI Design guidelines, Chakra UI

Backend: MySQL

Integration: NodeJS (MySQL Package)

Project Analysis:

The major components of the app are the Customer module, Desk Agent module and the Delivery Agent module.

Customer Module:

- The module consists of the cart, sign in, signup, orders, place order, menu, address routes of the app
- Cart: This route consists of the Items the user has added to the cart from the menu and their respective quantity and prices
- Sign In: This route consists of the page for the user sign in, it takes in the phone no and OTP to cross check it against the DB for login verification
- Sign Up: This route takes the user to the signup page where the user is asked to enter the phone no and address
- Orders: This route consists of all the orders made by the logged in user
- Address: This route displays the delivery address of the user which can also be edited by him/her
- Menu: This route Displays all the items available in the menu to the user
- Place Order: this route is just used to send the request to the DB and the backend app to place order

Desk Agent:

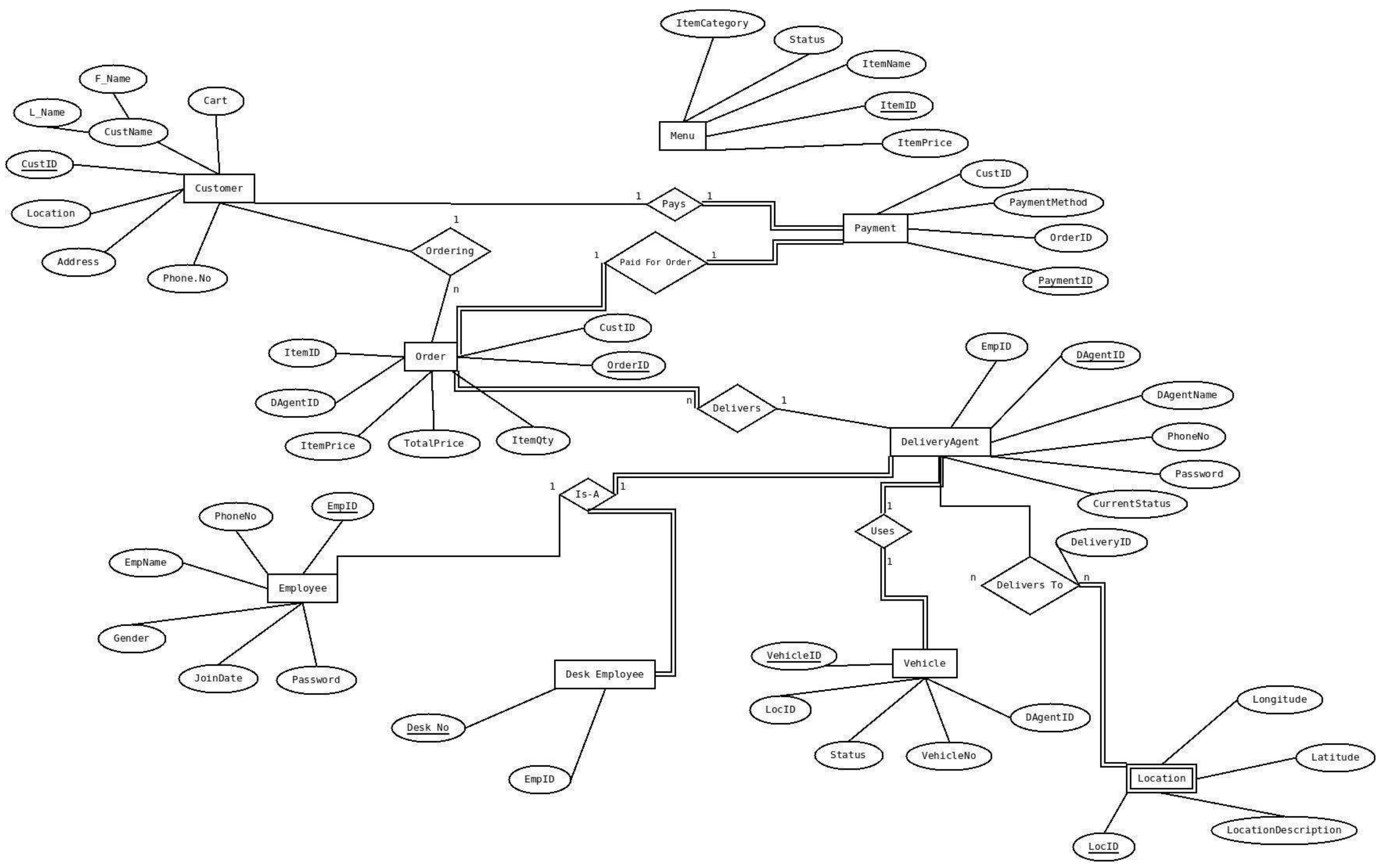
- This module is the Dash board of the Desk Agent who assigns the Orders to the Delivery Agents
- The module displays a Dashboard of the available and unallocated orders to the Desk Employee
- The Employee can also edit the Menu using the **Edit Menu** sub-module.

Delivery Agent:

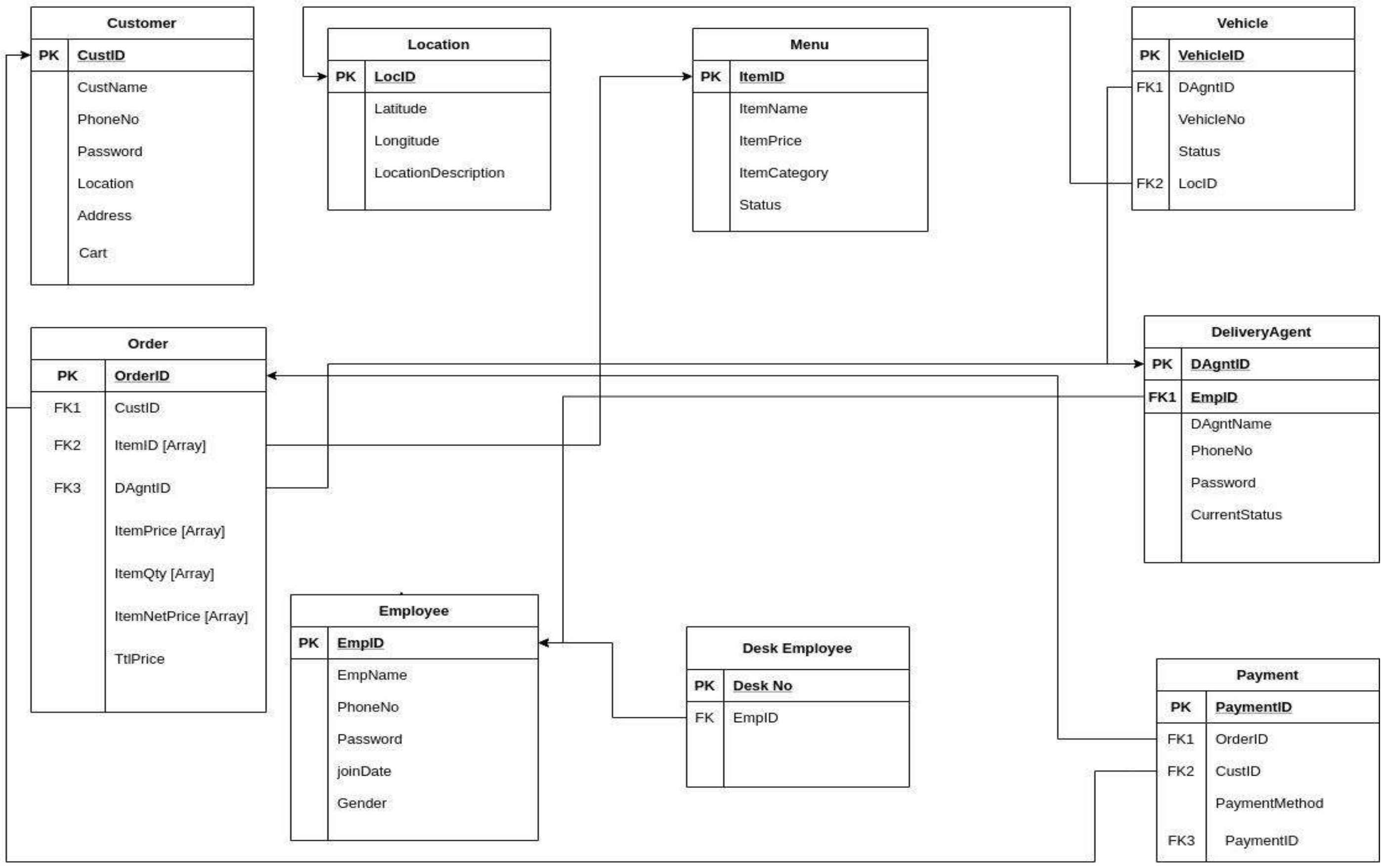
- This module is presented to the Delivery Agent
- This module gets the order allocated to the logged in
- Once an order has been delivered the delivery agent clicks on the delivered button.
- This action updates the DB and makes the delivery agent available for future order allocations.

Project Design:

The Initial ER Diagram



The Initial Schema



Customer

PK	**CustID**
	CustName
	PhoneNo
	Password
	Location
	Address
	Cart

Location

PK	**LocID**
	Latitude
	Longitude
	LocationDescription

Menu

PK	**ItemID**
	ItemName
	ItemPrice
	ItemCategory
	Status

Vehicle

PK	**VehicleID**
FK1	DAgntID
	VehicleNo
	Status
FK2	LocID

Order

PK	**OrderID**
FK1	CustID
FK2	ItemID [Array]
FK3	DAgntID
	ItemPrice [Array]
	ItemQty [Array]
	ItemNetPrice [Array]
	TtlPrice

DeliveryAgent

PK	**DAgntID**
FK1	**EmpID**
	DAgntName
	PhoneNo
	Password
	CurrentStatus

Employee

PK	**EmpID**
	EmpName
	PhoneNo
	Password
	joinDate
	Gender

Desk Employee

PK	**Desk No**
FK	EmpID

Payment

PK	**PaymentID**
FK1	OrderID
FK2	CustID
	PaymentMethod
FK3	PaymentID

Normalization:

Adding all attributes to a single table

AttribTable(CustID,CustName,LocID,Phone.no,Cart,ItemID,DAgent,ItemPrice,TotalPrice,ItemQty, OrderID,EmpID,Phone.no,EmpName,Gender,JoinDate,Password,Desk.no,EmpID,DAgentID,DAgentName,Phone.no,Password,CurrentStatus,DeliveryID,Address,Landmark,VehicleID,Currentstatus,PaymentID,PaymentMethod,PaymentAmt,ItemName,ItemStatus)

Fucntional Dependencies:

CustID -> CustName,LocID,Phone.no,Cart

CustID,OrderID -> ItemID,DAgentID,ItemPrice,TotalPrice,ItemQty

EmpID -> EmpName,Phone,Gender,JoinDate, Password,Deskno,CurrentStatus,DAgentID,DAgentName

LocID -> Landmark,Address

CustID,OrderID -> PaymentID

DAgentID -> DAgentName,Password,EmpID,Phone,CurrentStatus

PaymentID -> PaymentMethod, PaymentAmt

ItemID -> ItemName,ItemPrice,ItemStatus

EmpName -> DAgentName

Phone -> Phone

Password -> Password

OrderID -> DeliveryID

DeliveryID -> DAgentID

VehicleID -> LocID

DAgentID -> LocID

Removing Multivalued attributes:

- The cart attribute in the Customer entity is multivalued as it contains the contents of the customer's cart
- We add Cart as a separate entity with CartID and CustID as Primary key

Specialisation:

- The Employee entity can be specialised by splitting it into two types of employees and having unique attributes to each
- Here we split the Entities into DeskEmployee and DeliveryAgent

1NF :

Customer(CustID,LocID,Phone,CustName,CartNo)

Cart(CartID, CustID, ItemID, ItemQty)

Order(OrderID, DAgentID, ItemPrice, TotalPrice, ItemQty, ItemID, CustID)

Employee(Phone,EmpName,Gender,JoinDate,Password,EmpID)

DeskEmployee(Deskno,EmpID)

DeliveryAgent(EmpID,DAgentID,DAgentName,Phone,Password,CurrentStatus)

Location(LocID, Landmark, Address)

Delivery(DeliveryID,DAgentID,OrderID,LocID)

Payment(PaymentID, PaymentAmt,PaymentID, PaymentMethod,OrderID,CustID)

Menu(ItemID,ItemName,ItemPrice,ItemStatus)

Vehicle(VehicleID,LocID,CurrentStatus, DAgentID)

Partial Dependencies:

EmpID -> DAgentName,Phone,Password

CustID -> ItemID, ItemQty

Removing Extraneous Attributes:

- The attributes DAgentId, DAgentName, Password, Phone are just copies of the EmpID, EmpName, Phone,Password in the Employee table
- The Primary key DAgentID is dependent on EmpID
- So we remove all these attributes out of the table and just have EmpID and CurrentStatus in the table

Replacing DAgentID with EmpID:

- As the attribute DAgentID is removed from the schema we have to replace it's occurrences with EmpID

2NF:

Customer(CustID,LocID,Phone,CustName,CartNo)

Cart(CustID, ItemID, ItemQty)

Order(OrderID, ItemPrice, TotalPrice, ItemQty, ItemID, CustID)

Employee(Phone,EmpName,Gender,JoinDate,Password,EmpID)

DeskEmployee(Deskno,EmpID)

DeliveryAgent(EmpID,Phone,Password,CurrentStatus)

Location(LocID, Landmark, Address)

Delivery(DeliveryID,EmpID,OrderID,LocID)

Payment(PaymentID, PaymentAmt,PaymentID, PaymentMethod,OrderID,CustID)

Menu(ItemID,ItemName,ItemPrice,ItemStatus)

Vehicle(VehicleID,LocID,CurrentStatus, EmpID)

Transitive Dependencies:

ItemID -> ItemQty => This applies in two Entities -> Order and Cart

EmpID -> CurrentStatus => We removed the attributes and added them to new entity

OrderID -> LocID,EmpID => DeliveryID Becomes an Extraneous attribute here (Transitivity Rule)

Changes:

- The Primary key of Cart is now both CustID and ItemID
- New Entity named Status is added to satisfy the EmpID -> CurrentStatus dependency
- OrderID is made the Primary Key of the Delivery entity
- To satisfy the ItemID -> ItemQty in the Order entity we add a new entity named OrderItems and Add ItemID,OrderID,ItemQty,ItemPrice to the entity and make ItemID and OrderID the primary key
- Since TotalPrice can be derived from ItemQty and ItemPrice It is also removed

3NF:

Customer(CustID,LocID,Phone,CustName,CartNo)

Cart(CustID, ItemID, ItemQty)

Order(OrderID, CustID)

OrderItems(OrderID, ItemPrice, ItemQty, ItemID)

Employee(Phone,EmpName,Gender,JoinDate,Password,EmpID)

DeskEmployee(Deskno,EmpID)

DeliveryAgent(EmpID,Phone,Password,CurrentStatus)

Location(LocID, Landmark, Address)

Delivery(EmpID,OrderID,LocID)

Payment(PaymentID, PaymentAmt,PaymentID, PaymentMethod,OrderID,CustID)

Menu(ItemID,ItemName,ItemPrice,ItemStatus)

Vehicle(VehicleID,LocID)

Status(CurrentStatus, EmpID)

Dependencies not satisfying BCNF:

OrderID -> PaymentID => in Payment entity

Changes:

- We added PaymentID as an attribute to the Order entity and removed OrderID and CustID from Payment entity
- Since the Dependencies satisfied by the status table is already satisfied by the DeliveryAgent entity, we remove the Status entity

Dependency Closure:

CustID -> CustName,LocID,Phone.no,Cart

CustID,OrderID -> ItemID,ItemPrice,ItemQty

EmpID -> EmpName,Phone,Gender,JoinDate, Password,Deskno,CurrentStatus

LocID -> Landmark,Address

CustID,OrderID -> PaymentID

PaymentID -> PaymentMethod, PaymentAmt

ItemID -> ItemName,ItemPrice,ItemStatus

DeliveryID -> EmpID

VehicleID -> LocID

EmpID -> LocID

OrderID -> PaymentID

ItemID -> ItemQty

EmpID -> CurrentStatus

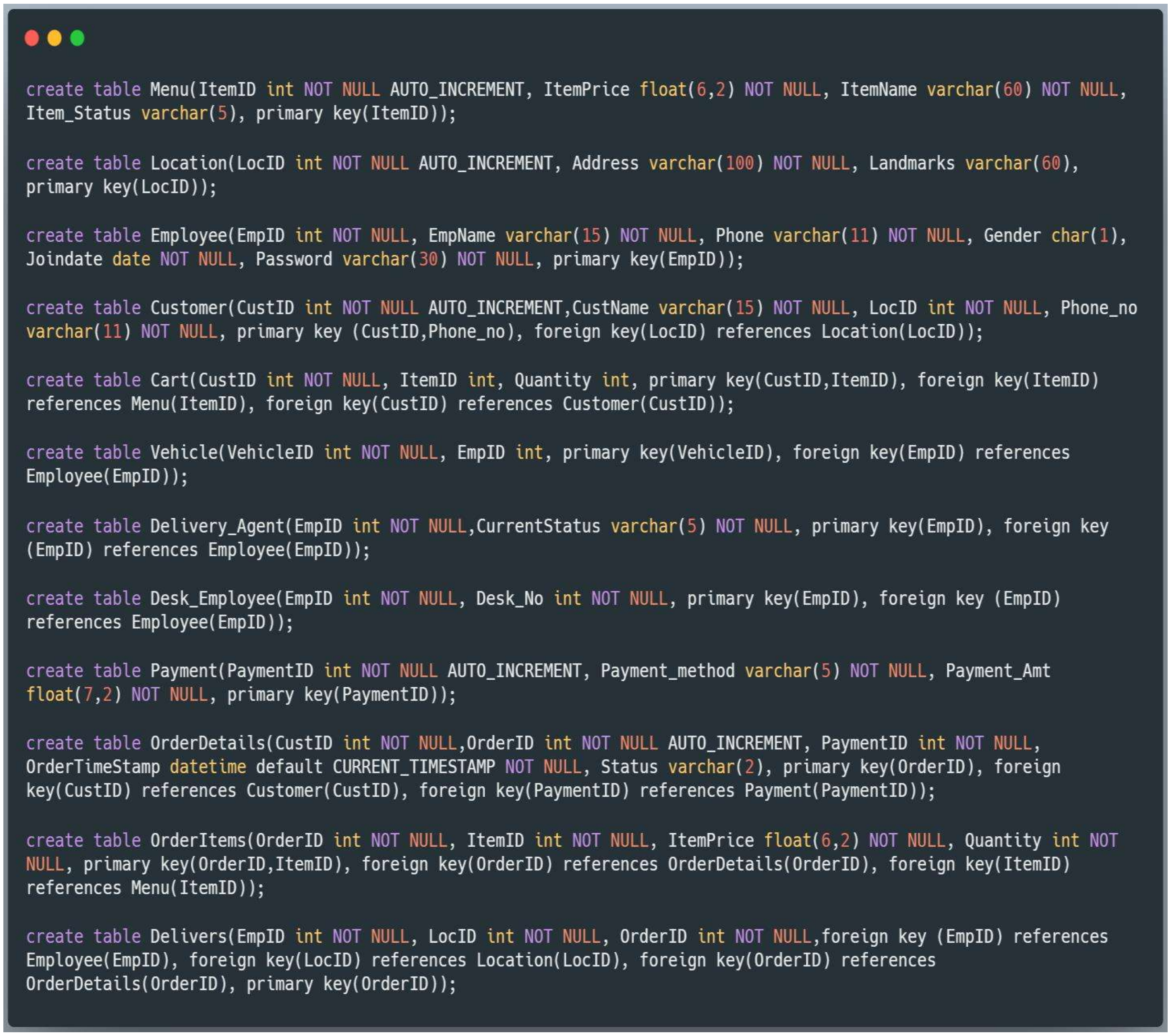
OrderID -> LocID,EmpID

CustID -> ItemID, ItemQty

- All the Dependencies are preserved and even after removing the Extraneous attributes the schema and dependencies are intact

Backend Design:

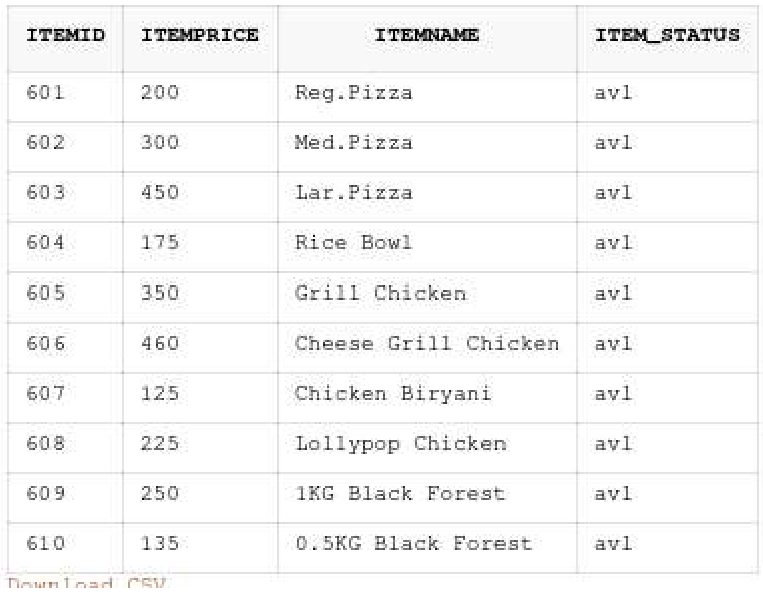
Table Creation Commands:(As in the integrated app)



```
create table Menu(ItemID int NOT NULL AUTO_INCREMENT, ItemPrice float(6,2) NOT NULL, ItemName varchar(60) NOT NULL,
Item_Status varchar(5), primary key(ItemID));

create table Location(LocID int NOT NULL AUTO_INCREMENT, Address varchar(100) NOT NULL, Landmarks varchar(60),
primary key(LocID));

create table Employee(EmpID int NOT NULL, EmpName varchar(15) NOT NULL, Phone varchar(11) NOT NULL, Gender char(1),
Joindate date NOT NULL, Password varchar(30) NOT NULL, primary key(EmpID));

create table Customer(CustID int NOT NULL AUTO_INCREMENT,CustName varchar(15) NOT NULL, LocID int NOT NULL, Phone_no
varchar(11) NOT NULL, primary key (CustID,Phone_no), foreign key(LocID) references Location(LocID));

create table Cart(CustID int NOT NULL, ItemID int, Quantity int, primary key(CustID,ItemID), foreign key(ItemID)
references Menu(ItemID), foreign key(CustID) references Customer(CustID));

create table Vehicle(VehicleID int NOT NULL, EmpID int, primary key(VehicleID), foreign key(EmpID) references
Employee(EmpID));

create table Delivery_Agent(EmpID int NOT NULL,CurrentStatus varchar(5) NOT NULL, primary key(EmpID), foreign key
(EmpID) references Employee(EmpID));

create table Desk_Employee(EmpID int NOT NULL, Desk_No int NOT NULL, primary key(EmpID), foreign key (EmpID)
references Employee(EmpID));

create table Payment(PaymentID int NOT NULL AUTO_INCREMENT, Payment_method varchar(5) NOT NULL, Payment_Amt
float(7,2) NOT NULL, primary key(PaymentID));

create table OrderDetails(CustID int NOT NULL,OrderID int NOT NULL AUTO_INCREMENT, PaymentID int NOT NULL,
OrderTimeStamp datetime default CURRENT_TIMESTAMP NOT NULL, Status varchar(2), primary key(OrderID), foreign
key(CustID) references Customer(CustID), foreign key(PaymentID) references Payment(PaymentID));

create table OrderItems(OrderID int NOT NULL, ItemID int NOT NULL, ItemPrice float(6,2) NOT NULL, Quantity int NOT
NULL, primary key(OrderID,ItemID), foreign key(OrderID) references OrderDetails(OrderID), foreign key(ItemID)
references Menu(ItemID));

create table Delivers(EmpID int NOT NULL, LocID int NOT NULL, OrderID int NOT NULL,foreign key (EmpID) references
Employee(EmpID), foreign key(LocID) references Location(LocID), foreign key(OrderID) references
OrderDetails(OrderID), primary key(OrderID));
```

Table Screenshots:

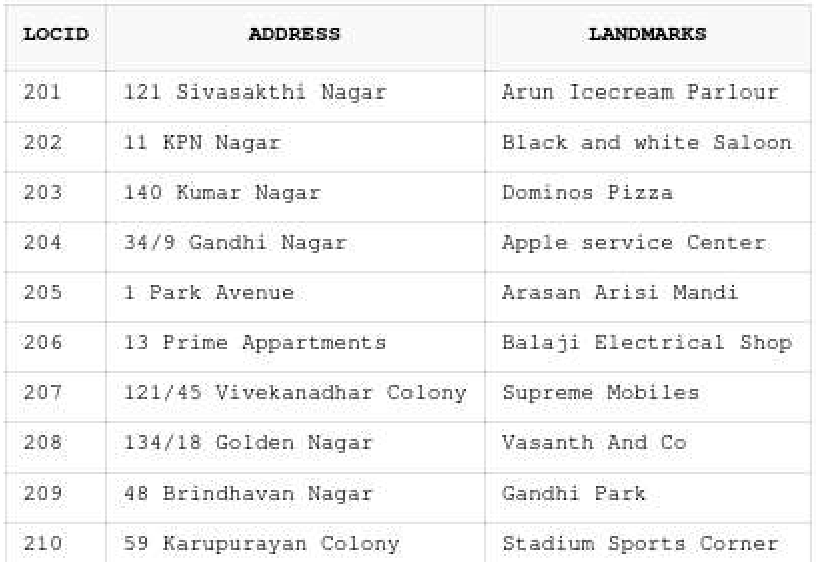
Menu



ITEMID	ITEMPRICE	ITEMNAME	ITEM_STATUS
601	200	Reg.Pizza	avl
602	300	Med.Pizza	avl
603	450	Lar.Pizza	avl
604	175	Rice Bowl	avl
605	350	Grill Chicken	avl
606	460	Cheese Grill Chicken	avl
607	125	Chicken Biryani	avl
608	225	Lollypop Chicken	avl
609	250	1KG Black Forest	avl
610	135	0.5KG Black Forest	avl

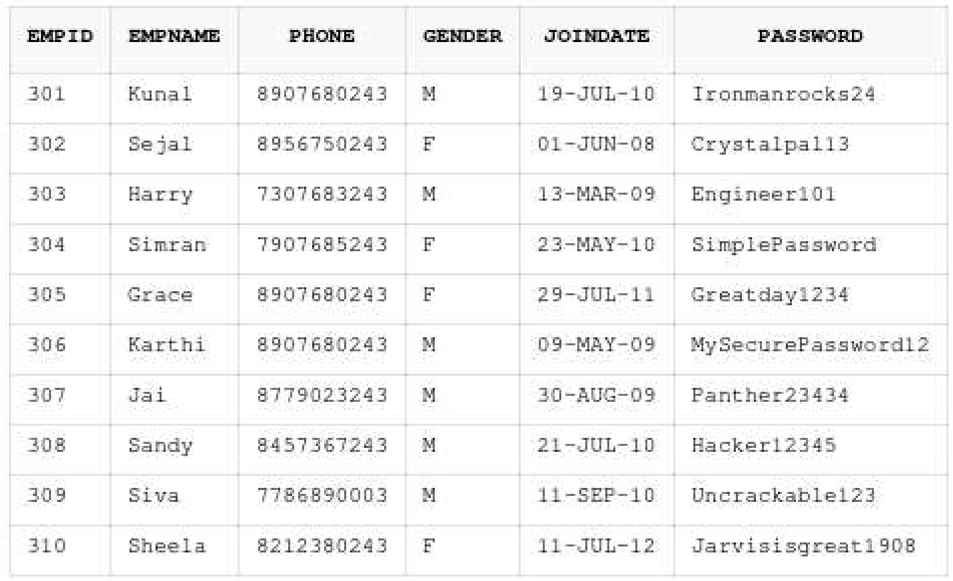
Download CSV

Location



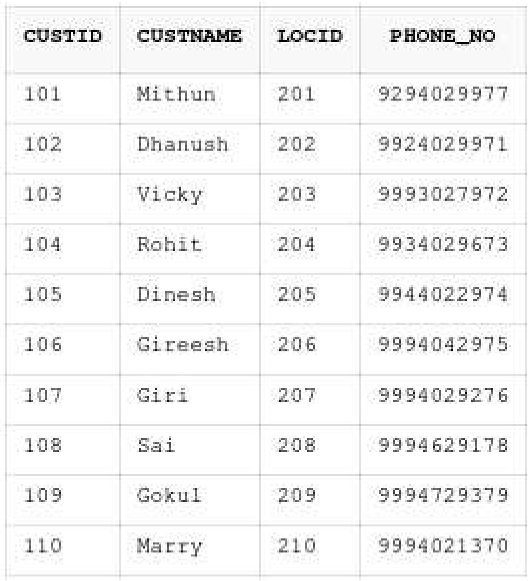
LOCID	ADDRESS	LANDMARKS
201	121 Sivasakthi Nagar	Arun Icecream Parlour
202	11 KPN Nagar	Black and white Saloon
203	140 Kumar Nagar	Dominos Pizza
204	34/9 Gandhi Nagar	Apple service Center
205	1 Park Avenue	Arasan Arisi Mandi
206	13 Prime Appartments	Balaji Electrical Shop
207	121/45 Vivekanadhar Colony	Supreme Mobiles
208	134/18 Golden Nagar	Vasanth And Co
209	48 Brindhavan Nagar	Gandhi Park
210	59 Karupurayan Colony	Stadium Sports Corner

Employee



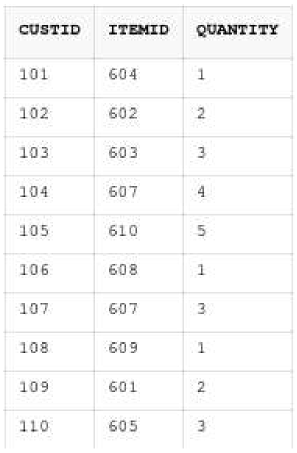
EMPID	EMPNAME	PHONE	GENDER	JOINDATE	PASSWORD
301	Kunal	8907680243	M	19-JUL-10	Ironmanrocks24
302	Sejal	8956750243	F	01-JUN-08	Crystalpal13
303	Harry	7307683243	M	13-MAR-09	Engineer101
304	Simran	7907685243	F	23-MAY-10	SimplePassword
305	Grace	8907680243	F	29-JUL-11	Greatday1234
306	Karthi	8907680243	M	09-MAY-09	MySecurePassword12
307	Jai	8779023243	M	30-AUG-09	Panther23434
308	Sandy	8457367243	M	21-JUL-10	Hacker12345
309	Siva	7786890003	M	11-SEP-10	Uncrackable123
310	Sheela	8212380243	F	11-JUL-12	Jarvisisgreat1908

Customer



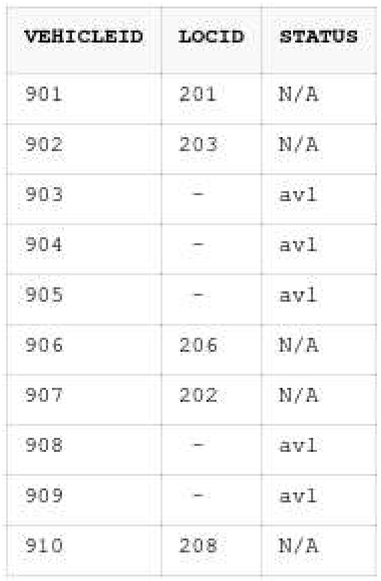
CUSTID	CUSTNAME	LOCID	PHONE_NO
101	Mithun	201	9294029977
102	Dhanush	202	9924029971
103	Vicky	203	9993027972
104	Rohit	204	9934029673
105	Dinesh	205	9944022974
106	Gireesh	206	9994042975
107	Giri	207	9994029276
108	Sai	208	9994629178
109	Gokul	209	9994729379
110	Marry	210	9994021370

Cart



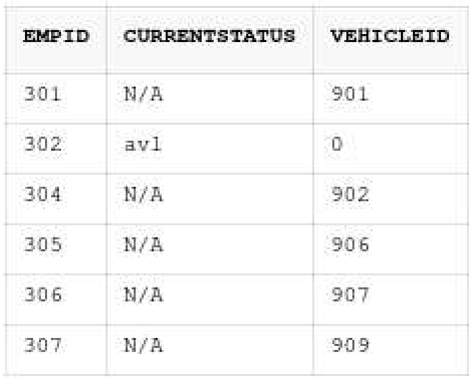
CUSTID	ITEMID	QUANTITY
101	604	1
102	602	2
103	603	3
104	607	4
105	610	5
106	608	1
107	607	3
108	609	1
109	601	2
110	605	3

Vehicle



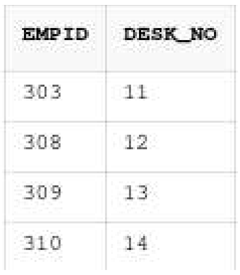
VEHICLEID	LOCID	STATUS
901	201	N/A
902	203	N/A
903	-	avl
904	-	avl
905	-	avl
906	206	N/A
907	202	N/A
908	-	avl
909	-	avl
910	208	N/A

Delivery_Agent



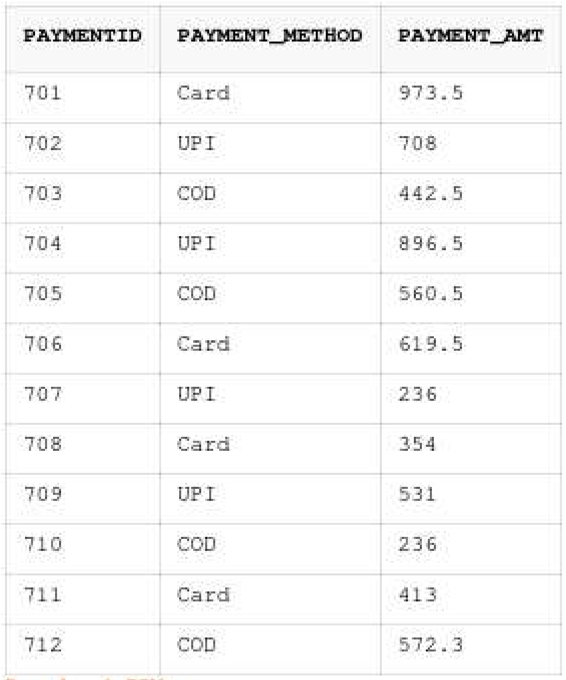
EMPID	CURRENTSTATUS	VEHICLEID
301	N/A	901
302	avl	0
304	N/A	902
305	N/A	906
306	N/A	907
307	N/A	909

Desk_Employee



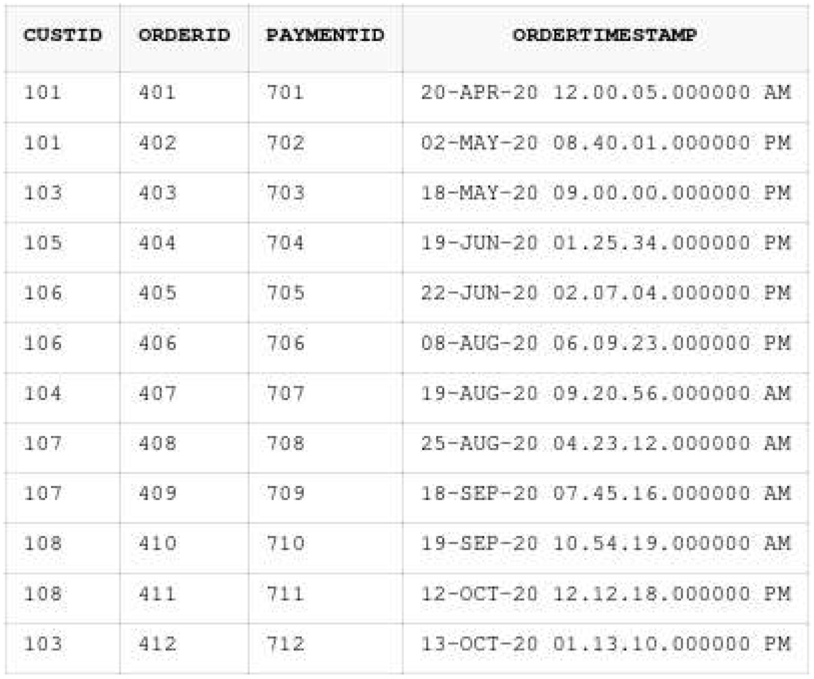
EMPID	DESK_NO
303	11
308	12
309	13
310	14

Payment



PAYMENTID	PAYMENT_METHOD	PAYMENT_AMT
701	Card	973.5
702	UPI	708
703	COD	442.5
704	UPI	896.5
705	COD	560.5
706	Card	619.5
707	UPI	236
708	Card	354
709	UPI	531
710	COD	236
711	Card	413
712	COD	572.3

OrderDetails



CUSTID	ORDERID	PAYMENTID	ORDERTIMESTAMP
101	401	701	20-APR-20 12.00.05.000000 AM
101	402	702	02-MAY-20 08.40.01.000000 PM
103	403	703	18-MAY-20 09.00.00.000000 PM
105	404	704	19-JUN-20 01.25.34.000000 PM
106	405	705	22-JUN-20 02.07.04.000000 PM
106	406	706	08-AUG-20 06.09.23.000000 PM
104	407	707	19-AUG-20 09.20.56.000000 AM
107	408	708	25-AUG-20 04.23.12.000000 AM
107	409	709	18-SEP-20 07.45.16.000000 AM
108	410	710	19-SEP-20 10.54.19.000000 AM
108	411	711	12-OCT-20 12.12.18.000000 PM
103	412	712	13-OCT-20 01.13.10.000000 PM

OrderItems

ORDERID	ITEMID	ITEMPRICE	QUANTITY
401	601	200	1
401	603	450	1
401	604	175	1
402	602	300	2
403	607	125	3
404	602	300	1
404	606	460	1
405	608	225	1
405	609	250	1
406	607	125	1
406	608	225	1
406	604	175	1
407	601	200	1
408	602	300	1
409	603	450	1
410	601	200	1
411	605	350	1
412	605	350	1
412	610	135	1

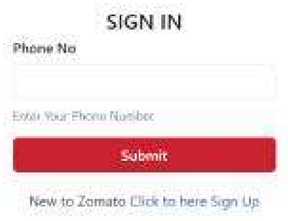
Delivers

EMPID	CURRENTSTATUS	VEHICLEID
301	N/A	901
302	avl	0
304	N/A	902
305	N/A	906
306	N/A	907
307	N/A	909

Frontend:

Tools Used

- VSCode and Sublime text 3 for text editors
- NodeJS with Axios and REST API for logics and the routes
- Material UI guidelines and Chakra UI for the UI guidelines



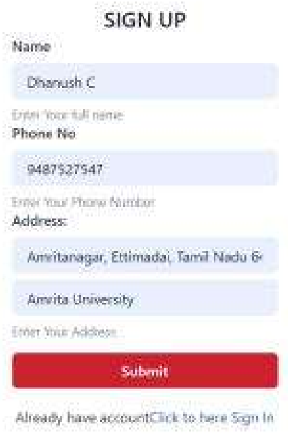
SIGN IN

Phone No

Enter Your Phone Number

Submit

New to Zomato Click to here Sign Up



SIGN UP

Name

Dhanush C

Enter Your full name

Phone No

9487527547

Enter Your Phone Number

Address:

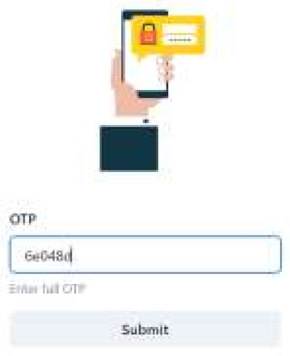
Amritanagar, Ettimadai, Tamil Nadu 6

Amrita University

Enter Your Address

Submit

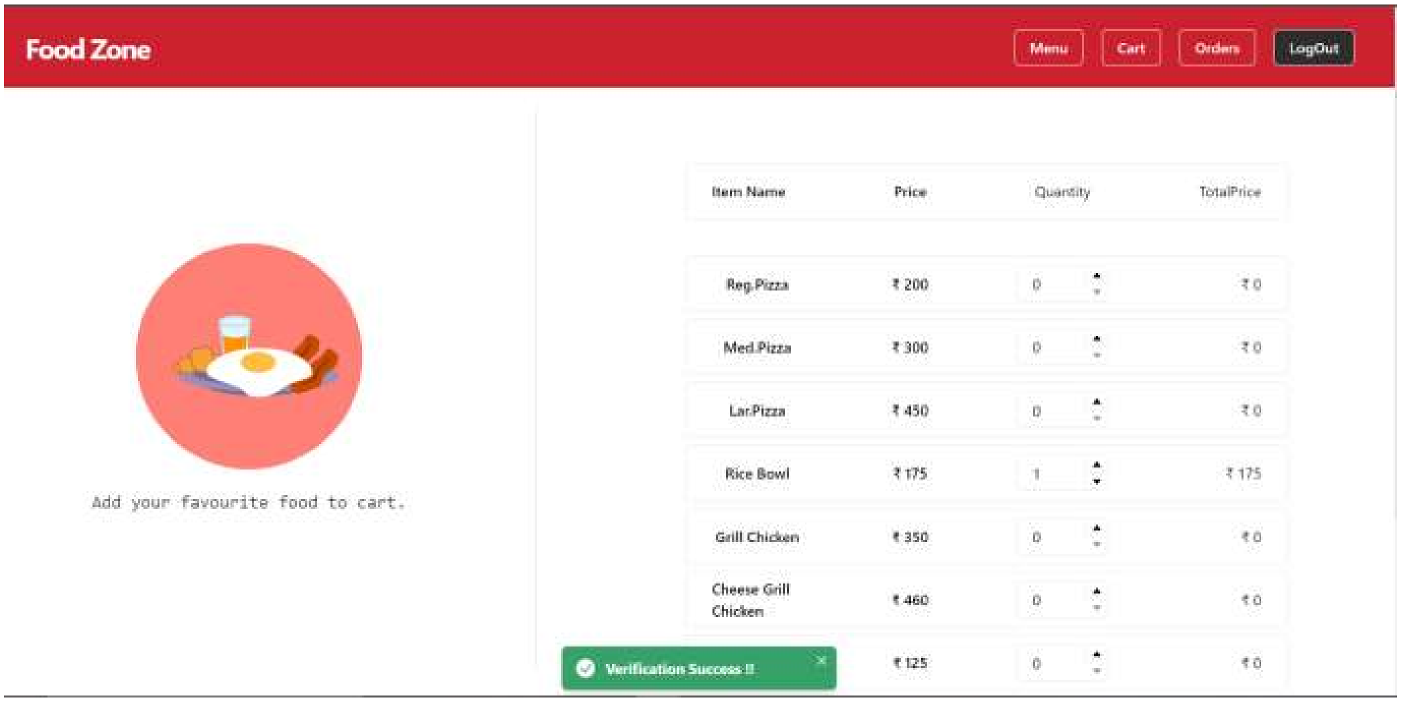
Already have accountClick to here Sign In



OTP

6e048d

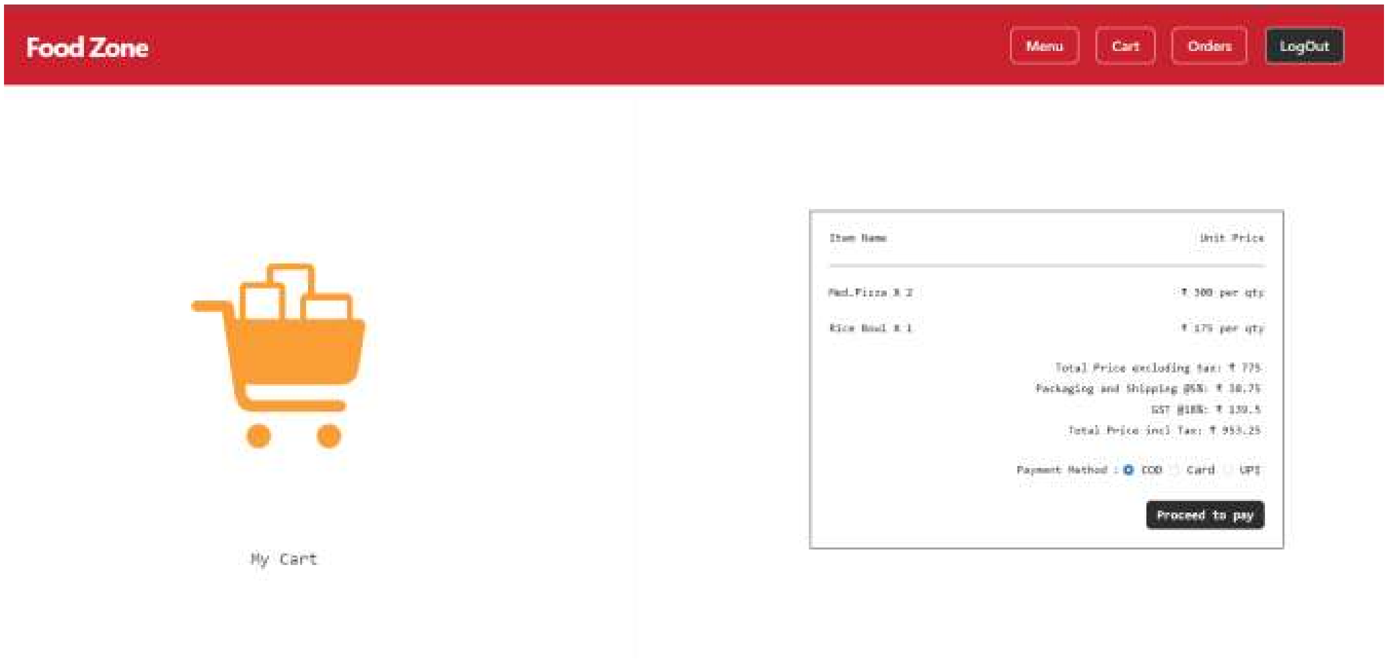
Enter full OTP



Add your favourite food to cart.

Item Name	Price	Quantity	TotalPrice
Reg.Pizza	₹ 200	0	₹ 0
Med.Pizza	₹ 300	0	₹ 0
Lar.Pizza	₹ 450	0	₹ 0
Rice Bowl	₹ 175	1	₹ 175
Grill Chicken	₹ 350	0	₹ 0
Cheese Grill Chicken	₹ 460	0	₹ 0
	₹ 125	0	₹ 0

Verification Success !!



Food Zone

Menu · Cart · Orders · LogOut

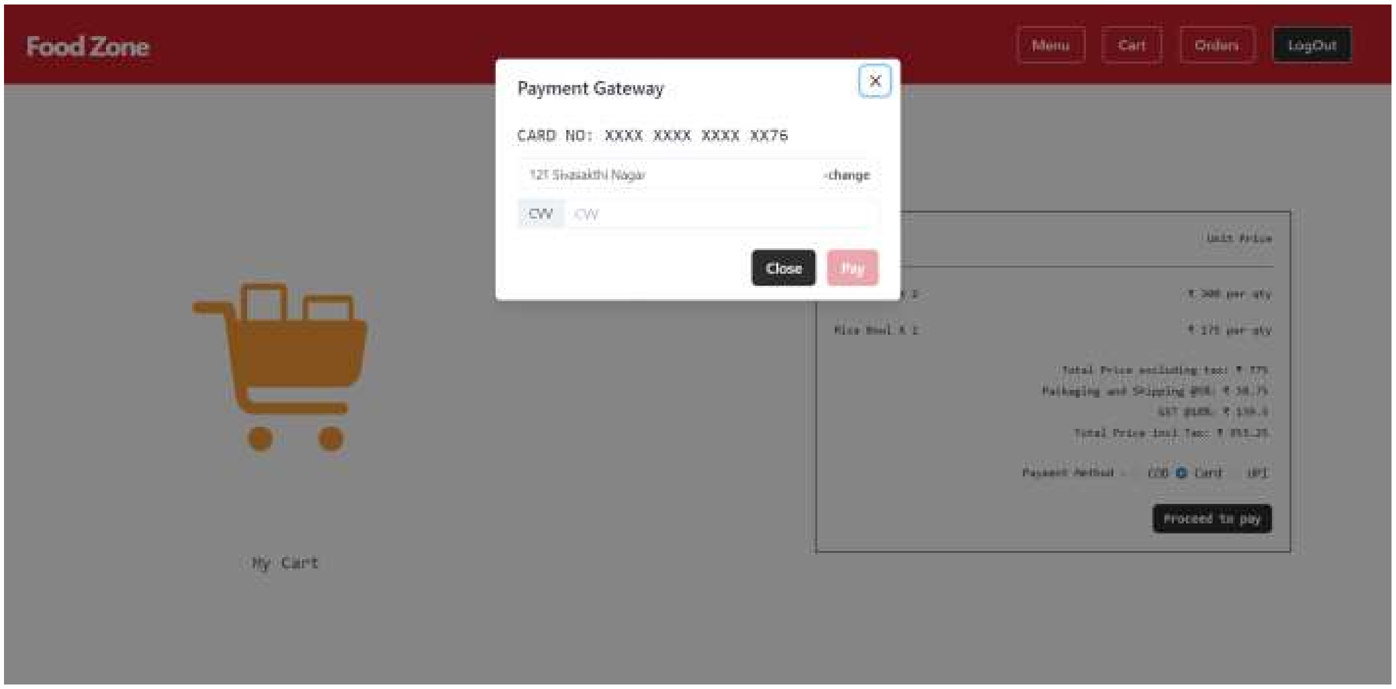
My Cart

Item Name	Unit Price
Med_Pizza X 2	₹ 300 per qty
Rice Bowl X 1	₹ 175 per qty

Total Price excluding tax: ₹ 775
Packaging and Shipping @5%: ₹ 38.75
GST @18%: ₹ 139.5
Total Price incl Tax: ₹ 953.25

Payment Method : [✓ COD] Card UPI

Proceed to pay



Food Zone

Menu · Cart · Orders · LogOut

Payment Gateway

CARD NO: XXXX XXXX XXXX XX76

121 Sivasakthi Nagar -change

CVV: CVV

Close · Pay

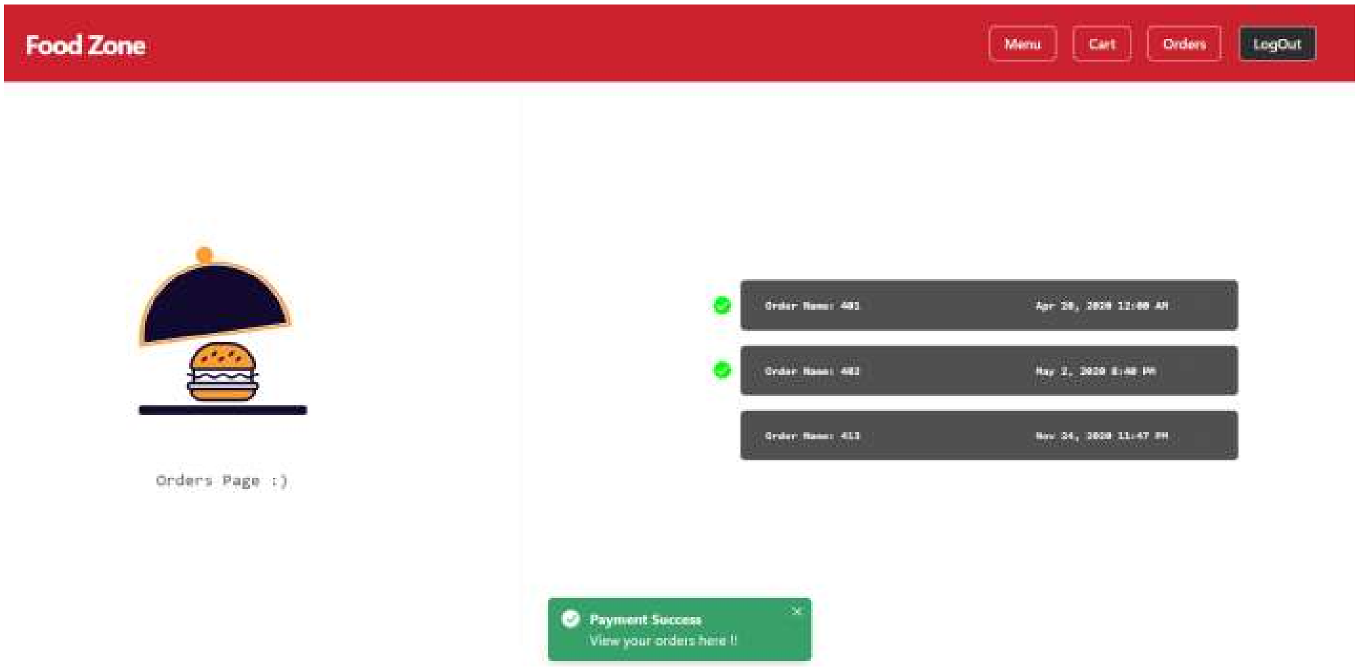
My Cart

Item Name	Unit Price
Med_Pizza X 2	₹ 300 per qty
Rice Bowl X 1	₹ 175 per qty

Total Price excluding tax: ₹ 775
Packaging and Shipping @5%: ₹ 38.75
GST @18%: ₹ 139.5
Total Price incl Tax: ₹ 953.25

Payment Method : COD [✓ Card] UPI

Proceed to pay

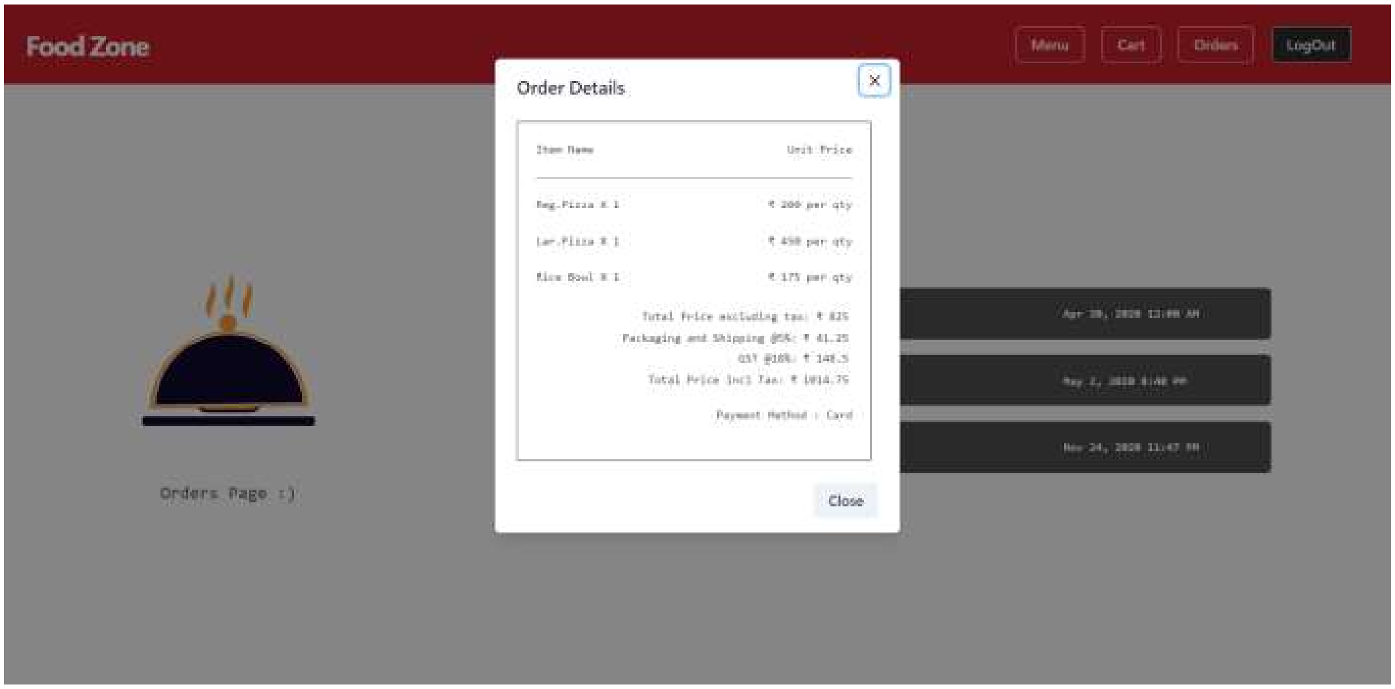


Order Name: 401 Apr 20, 2020 12:00 AM

Order Name: 402 May 2, 2020 8:40 PM

Order Name: 413 Nov 24, 2020 11:47 PM

Payment Success
View your orders here !!



Order Details

Item Name	Unit Price
Reg.Pizza X 1	₹ 200 per qty
Lar.Pizza X 1	₹ 450 per qty
Rice Bowl X 1	₹ 175 per qty

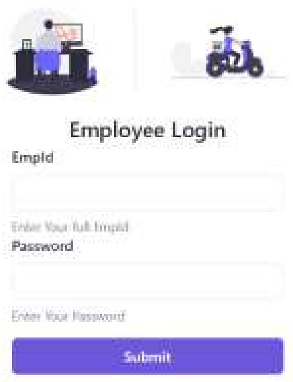
Total Price excluding tax: ₹ 825
Packaging and Shipping @5%: ₹ 41.25
GST @18%: ₹ 148.5
Total Price Incl Tax: ₹ 1014.75

Payment Method : Card

Apr 20, 2020 12:00 AM

May 2, 2020 8:40 PM

Nov 24, 2020 11:47 PM


Employee Login

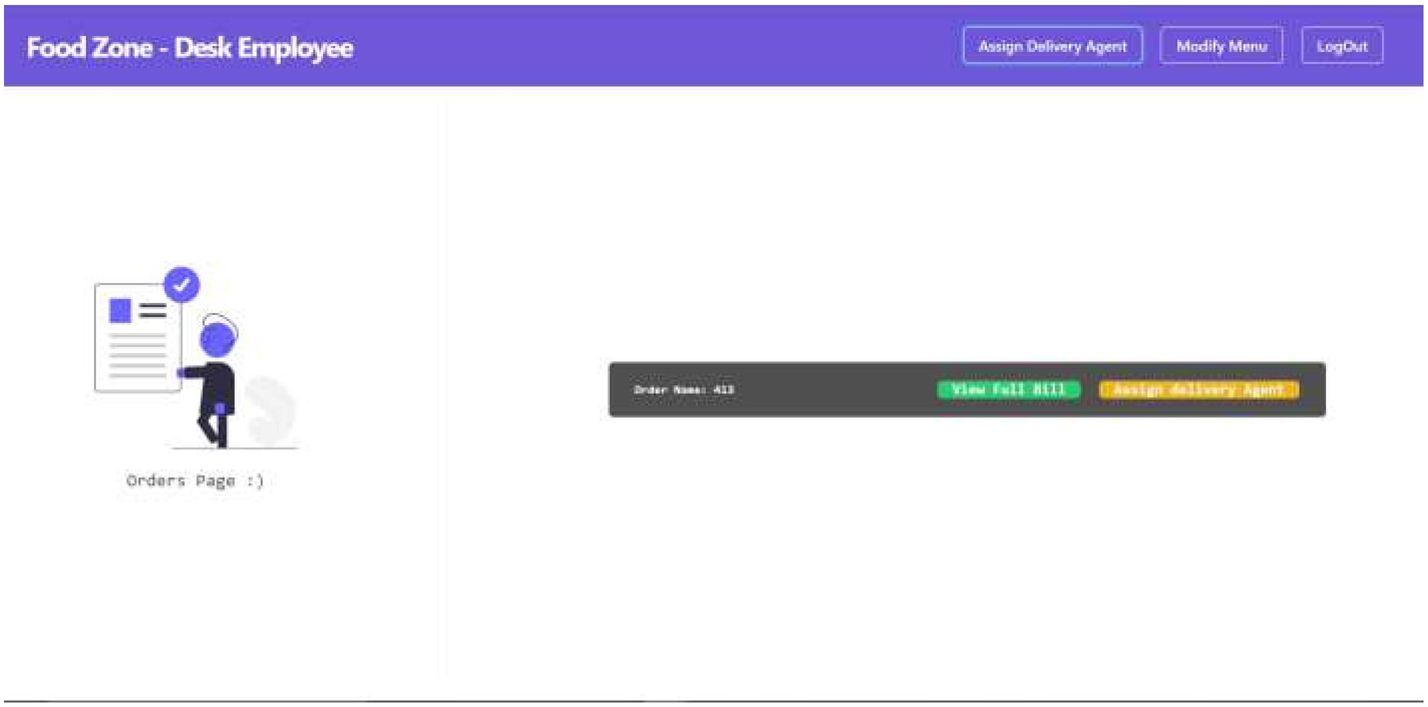
EmpId

Enter Your full EmpId

Password

Enter Your Password

Submit


Food Zone - Desk Employee

Assign Delivery Agent

Modify Menu

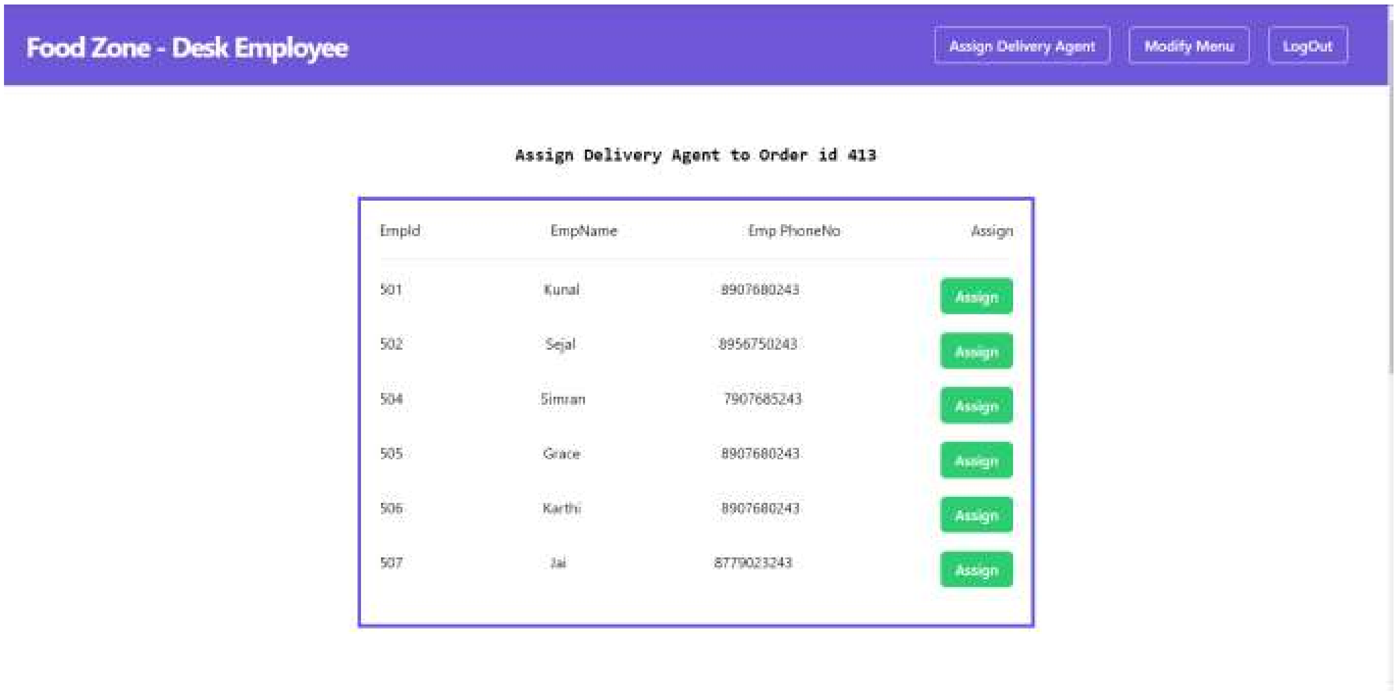
LogOut

Orders Page :)

Order Name: 413

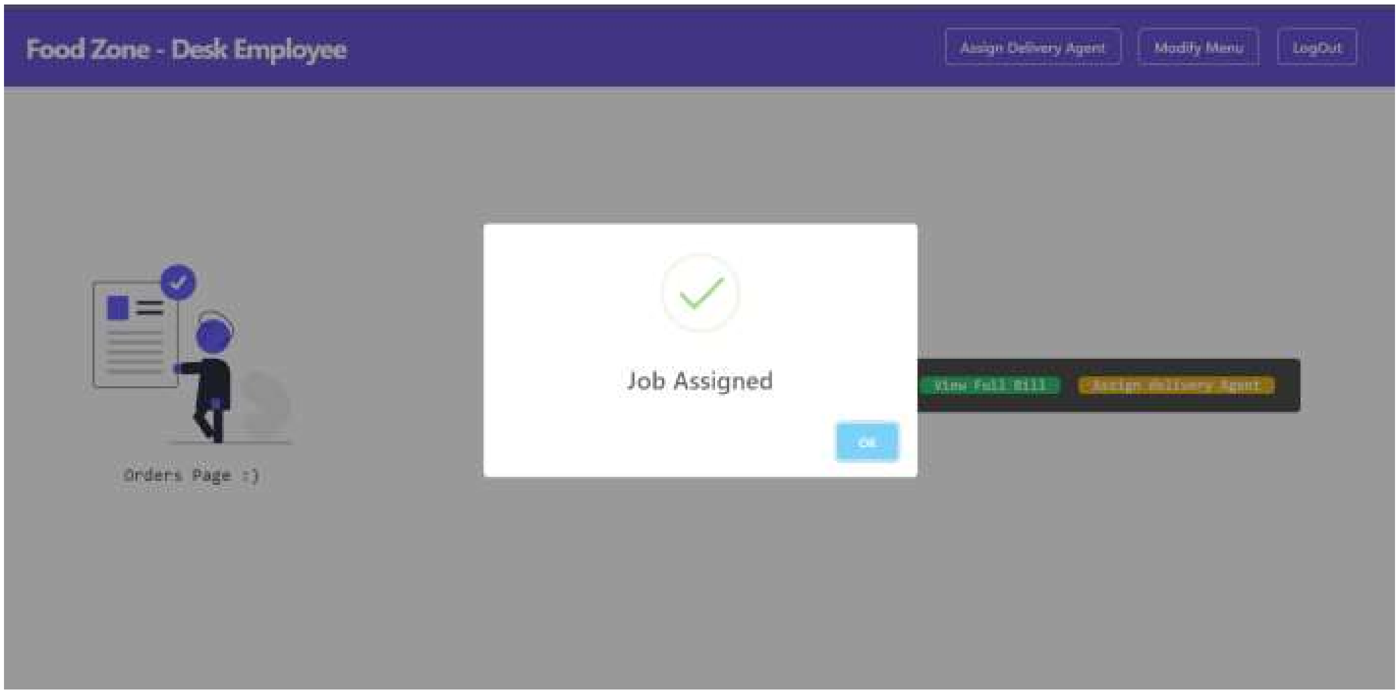
View Full Bill

Assign delivery Agent

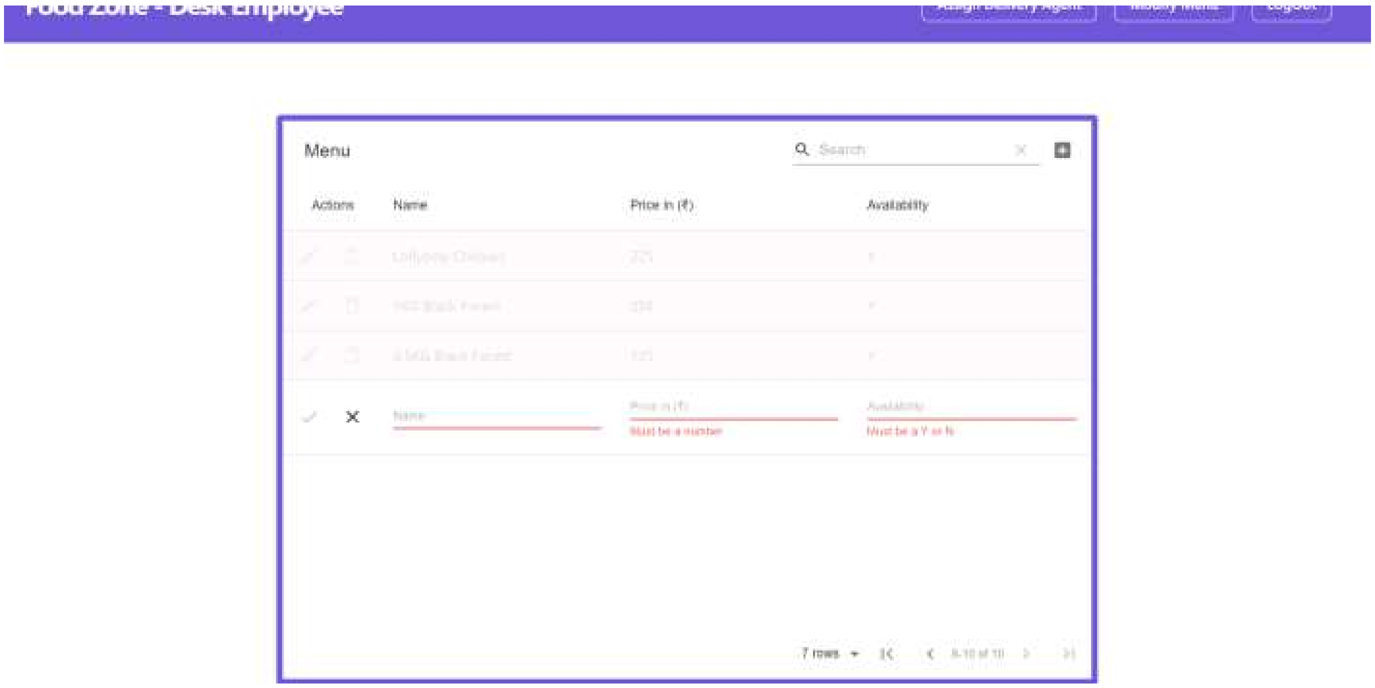


Assign Delivery Agent to Order id 413

EmpId	EmpName	Emp PhoneNo	Assign
501	Kunal	8907680243	Assign
502	Sejal	8956750243	Assign
504	Simran	7907685243	Assign
505	Grace	8907680243	Assign
506	Karthi	8907680243	Assign
507	Jai	8779023243	Assign



Job Assigned



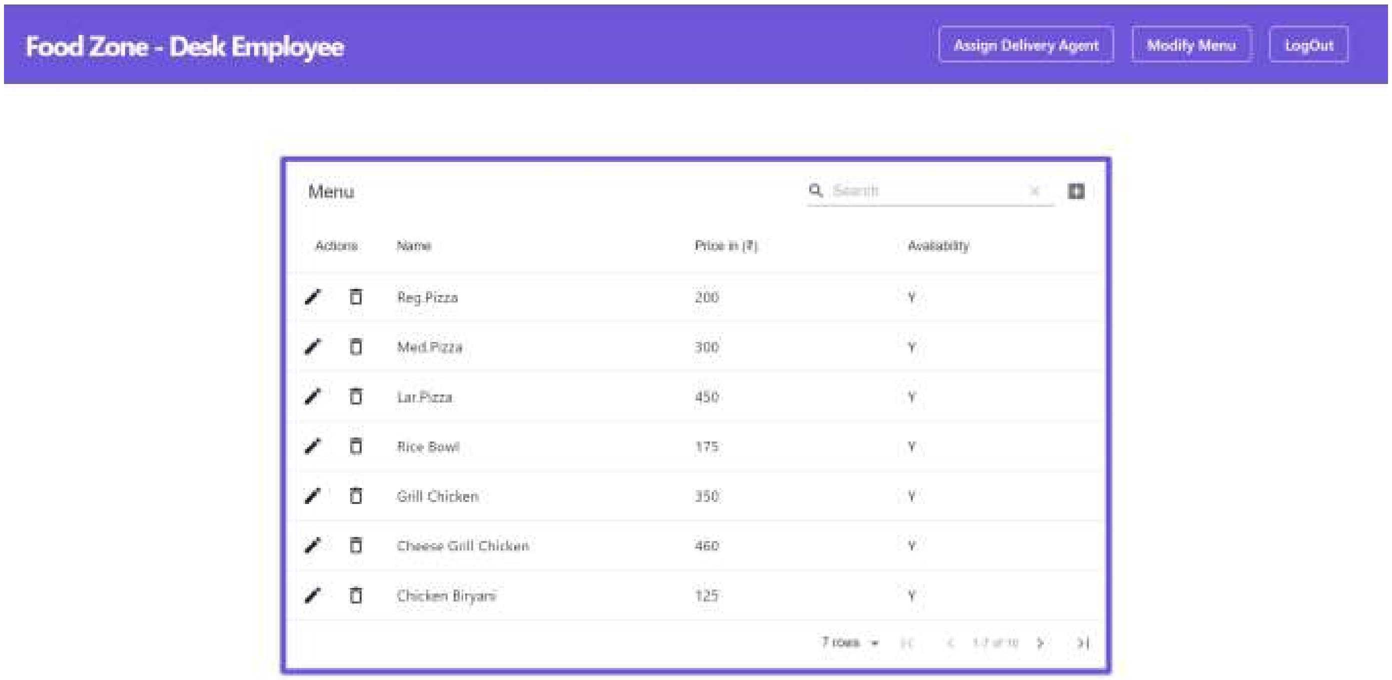
Food Zone - Desk Employee

Assign Delivery Agent | Modify Menu | LogOut

Menu

Actions	Name	Price in (₹)	Availability
	Lollypop Chicken	225	Y
	1KG Black Forest	350	Y
	0.5KG Black Forest	125	Y
	Name	Price in (₹) Must be a number	Availability Must be a Y or N

7 rows 8-10 of 10



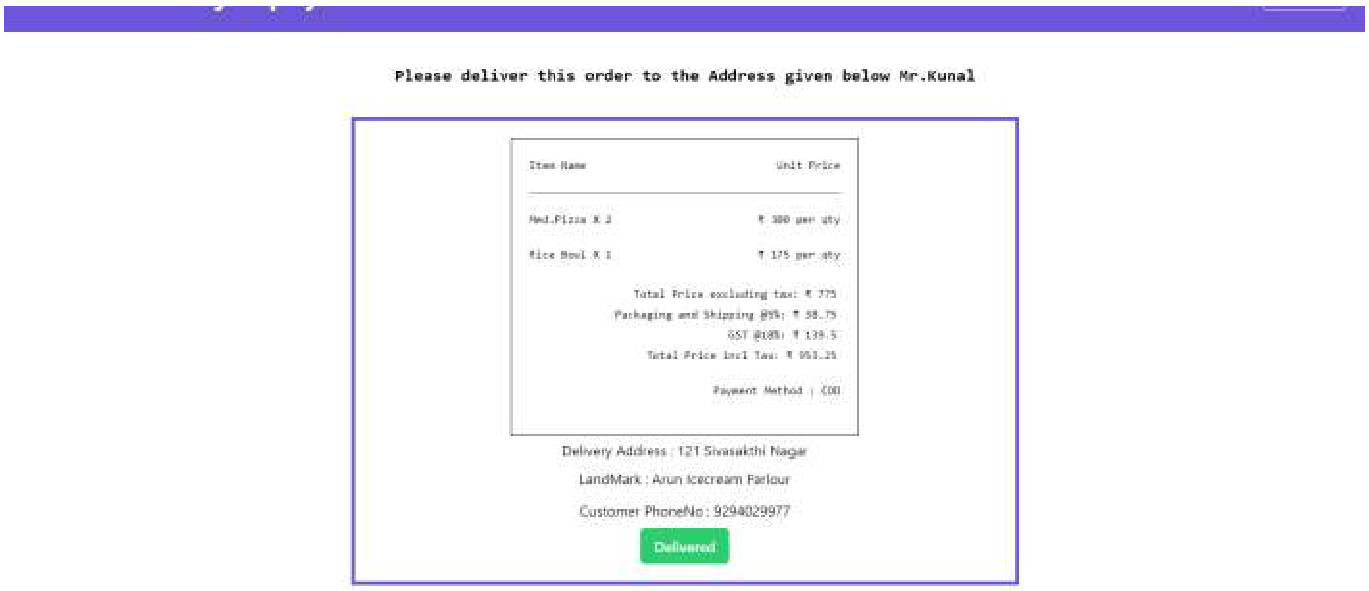
Food Zone - Desk Employee

Assign Delivery Agent | Modify Menu | LogOut

Menu

Actions	Name	Price in (₹)	Availability
	Reg.Pizza	200	Y
	Med.Pizza	300	Y
	Lar.Pizza	450	Y
	Rice Bowl	175	Y
	Grill Chicken	350	Y
	Cheese Grill Chicken	460	Y
	Chicken Biryani	125	Y

7 rows 1-7 of 10



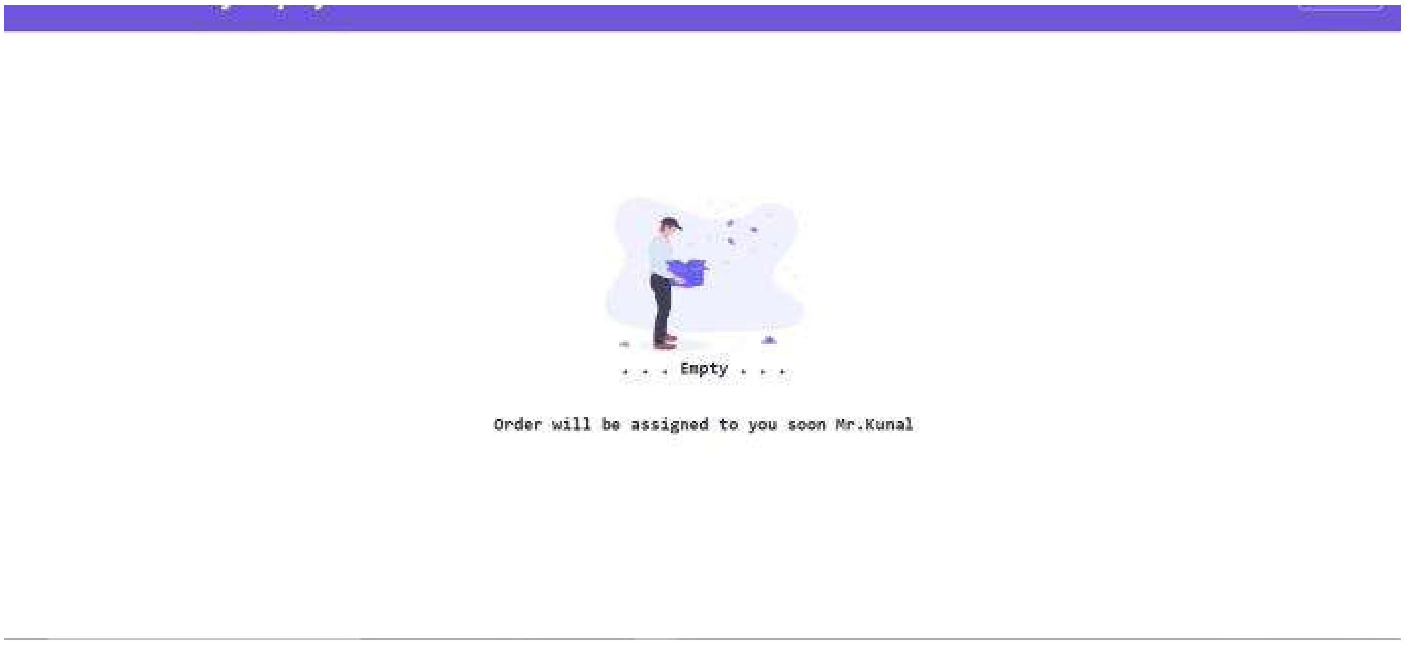
Please deliver this order to the Address given below Mr.Kunal

```
Item Name                         Unit Price
--------------------------------------------
Med.Pizza X 2                 ₹ 300 per qty

Rice Bowl X 1                 ₹ 175 per qty

         Total Price excluding tax: ₹ 775
      Packaging and Shipping @5%: ₹ 38.75
                      GST @18%: ₹ 139.5
          Total Price incl Tax: ₹ 953.25

                    Payment Method : COD
```

Delivery Address : 121 Sivasakthi Nagar

LandMark : Arun Icecream Parlour

Customer PhoneNo : 9294029977



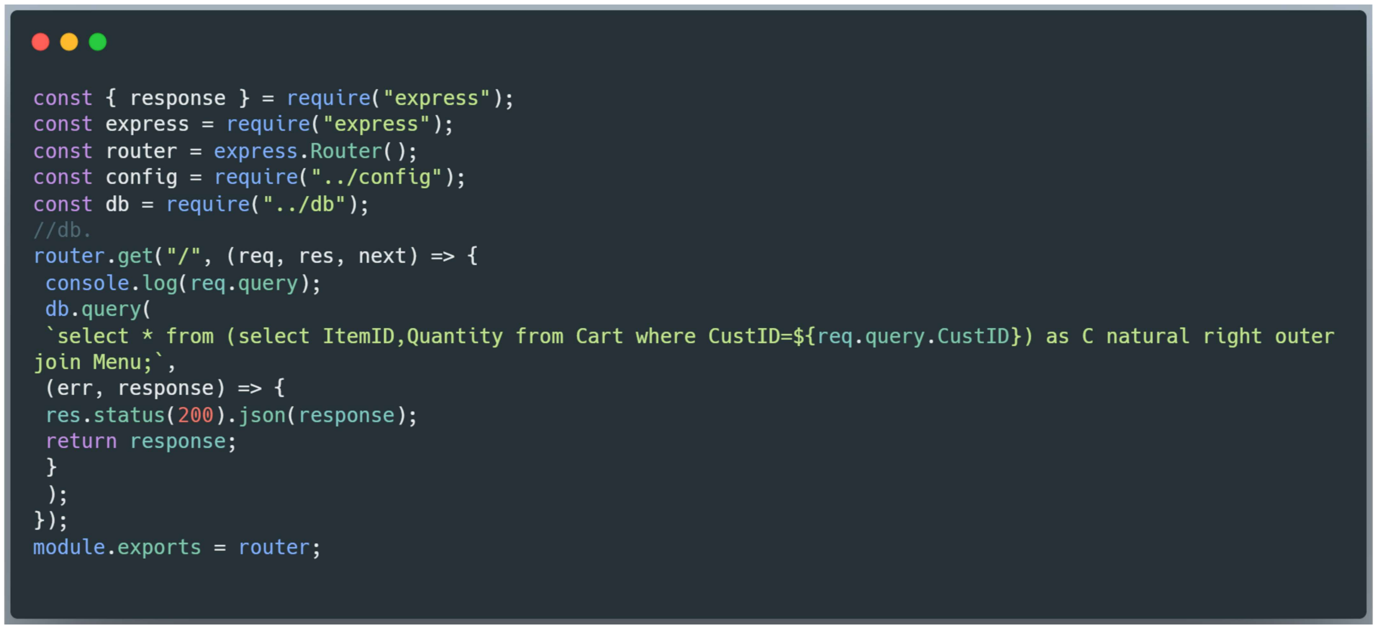
. . . Empty . . .

Order will be assigned to you soon Mr.Kunal

DATABASE Connectivity:

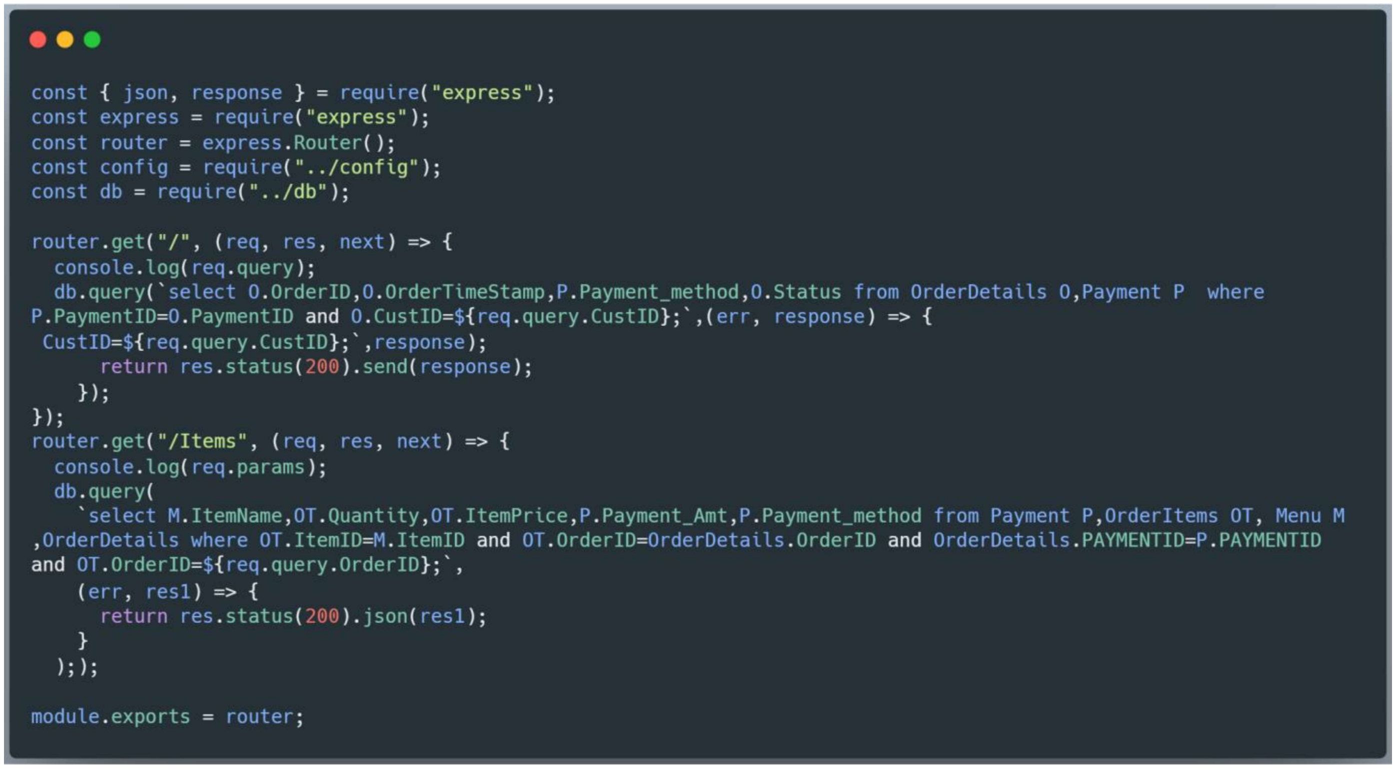
SAMPLE CODE

<u>MENU ROUTE:</u>



```
const { response } = require("express");
const express = require("express");
const router = express.Router();
const config = require("../config");
const db = require("../db");
//db.
router.get("/", (req, res, next) => {
 console.log(req.query);
 db.query(
 `select * from (select ItemID,Quantity from Cart where CustID=${req.query.CustID}) as C natural right outer
join Menu;`,
 (err, response) => {
 res.status(200).json(response);
 return response;
 }
 );
});
module.exports = router;
```

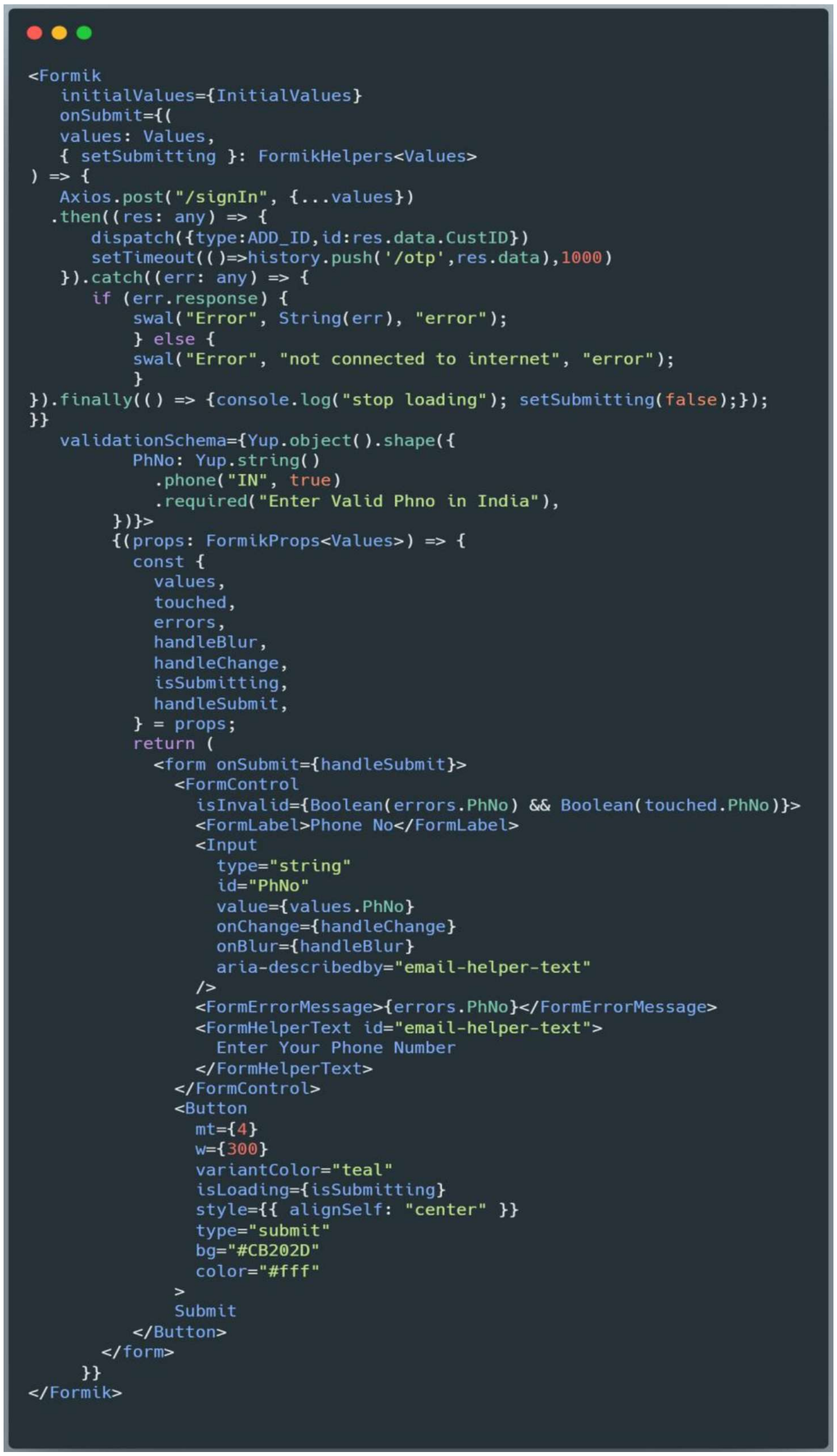
<u>My Order Route:</u>



```
const { json, response } = require("express");
const express = require("express");
const router = express.Router();
const config = require("../config");
const db = require("../db");

router.get("/", (req, res, next) => {
  console.log(req.query);
  db.query(`select O.OrderID,O.OrderTimeStamp,P.Payment_method,O.Status from OrderDetails O,Payment P  where
P.PaymentID=O.PaymentID and O.CustID=${req.query.CustID};`,(err, response) => {
 CustID=${req.query.CustID};`,response);
      return res.status(200).send(response);
    });
});
router.get("/Items", (req, res, next) => {
  console.log(req.params);
  db.query(
   `select M.ItemName,OT.Quantity,OT.ItemPrice,P.Payment_Amt,P.Payment_method from Payment P,OrderItems OT, Menu M
,OrderDetails where OT.ItemID=M.ItemID and OT.OrderID=OrderDetails.OrderID and OrderDetails.PAYMENTID=P.PAYMENTID
and OT.OrderID=${req.query.OrderID};`,
    (err, res1) => {
      return res.status(200).json(res1);
    }
  ););

module.exports = router;
```

SAMPLE UI CODE

Customer SignIn Screen



```
<Formik
   initialValues={InitialValues}
   onSubmit={(
   values: Values,
   { setSubmitting }: FormikHelpers<Values>
) => {
   Axios.post("/signIn", {...values})
  .then((res: any) => {
      dispatch({type:ADD_ID,id:res.data.CustID})
      setTimeout(()=>history.push('/otp',res.data),1000)
   }).catch((err: any) => {
      if (err.response) {
          swal("Error", String(err), "error");
          } else {
          swal("Error", "not connected to internet", "error");
          }
}).finally(() => {console.log("stop loading"); setSubmitting(false);});
}}
   validationSchema={Yup.object().shape({
          PhNo: Yup.string()
            .phone("IN", true)
            .required("Enter Valid Phno in India"),
        })}>
        {(props: FormikProps<Values>) => {
          const {
            values,
            touched,
            errors,
            handleBlur,
            handleChange,
            isSubmitting,
            handleSubmit,
          } = props;
          return (
            <form onSubmit={handleSubmit}>
              <FormControl
                isInvalid={Boolean(errors.PhNo) && Boolean(touched.PhNo)}>
                <FormLabel>Phone No</FormLabel>
                <Input
                  type="string"
                  id="PhNo"
                  value={values.PhNo}
                  onChange={handleChange}
                  onBlur={handleBlur}
                  aria-describedby="email-helper-text"
                />
                <FormErrorMessage>{errors.PhNo}</FormErrorMessage>
                <FormHelperText id="email-helper-text">
                  Enter Your Phone Number
                </FormHelperText>
              </FormControl>
              <Button
                mt={4}
                w={300}
                variantColor="teal"
                isLoading={isSubmitting}
                style={{ alignSelf: "center" }}
                type="submit"
                bg="#CB202D"
                color="#fff"
              >
              Submit
          </Button>
       </form>
     }}
</Formik>
```

CONCLUSION

This project helped us gain the experience of developing a full-stack app which reflects our idea to solve a real-world problem.

But our App is not yet in its complete form. During our project tenure new ideas kept popping up. We hope to continue working on it as a healthy hobby to add more.

OUR COMPLETE CODE: https://github.com/Dhanushc00/DbmsCourseproject

REFERENCES

https://www.mysql.com/

https://reactjs.org/

https://material-table.com/#/

https://github.com/microsoft/TypeScript

https://nodejs.org/en/docs/

https://chakra-ui.com/

https://learning.postman.com/docs/publishing-your-api/documenting-your-api/

https://github.com/axios/axios

https://undraw.co/

https://lottiefiles.com/

https://formik.org/

https://www.npmjs.com/package/yup