GROCERY SHOP

Kunal Pasad – UID: 2021300093

Rahul Ruke – UID: 2021300104

M Reddy – UID: 20213000105

Adwait Purao – UID: 2021300101

Aim:

We aim to create an interface for a Grocery Shop. The interface will allow customers to log in buy items and make transactions.

Technologies to be used:

Frontend:

1) HTML

2) CSS

3) JavaScript

Backend:

1) MySQL / MongoDB

2) Python

Timeline:

02/10/22 - Creating basic structure of each page of the interface.

10/10/22 – Completing with Frontend

16/11/22- Completing with Backend

25/11/22- Final debugging and working.

Framework Of the project:

- A Landing/Information Page to display information about the Grocery

Shop.

- A landing page with option for customer and employee login.
- For customer login the page will be directed to another page containing

some suggested items and a search bar to search for the respective items.

- On this page the user can view and add items to his order cart and an

order button to direct towards the Payment page.

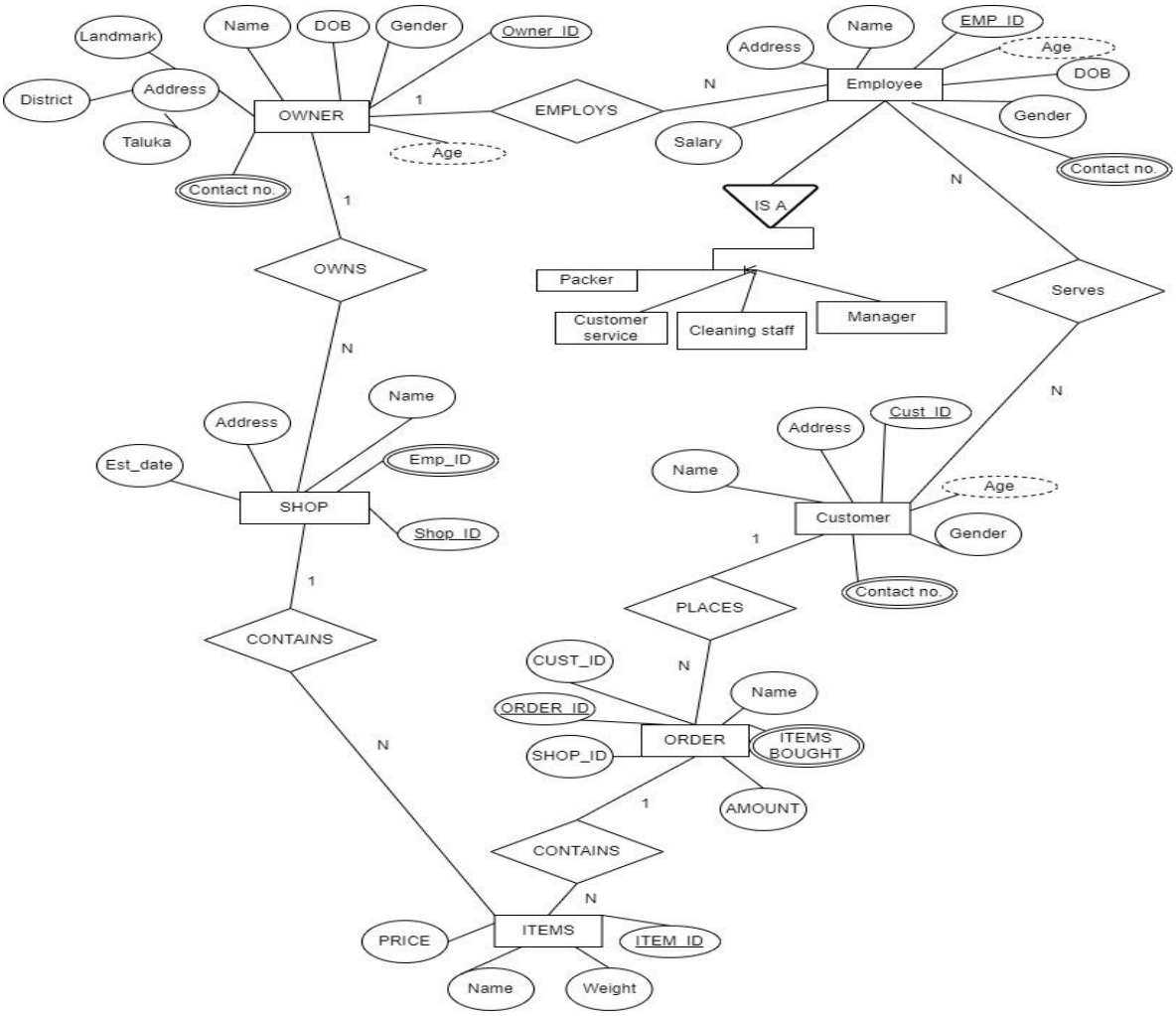
- The Payment page will take necessary details and confirm the order.
- Options to show profile of the customer/employee and options to update

the same.

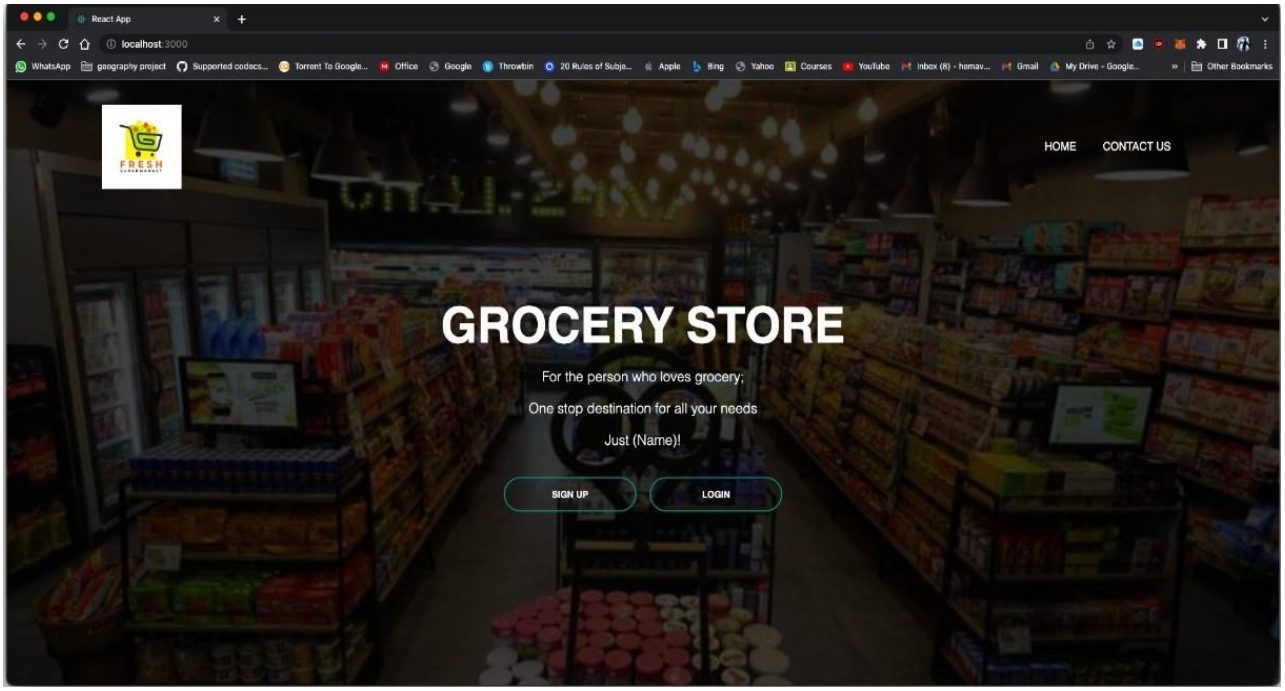
- Options for employees/customers to edit profile as per need.
- For the admin login, the page will be directed to another page showing

all the data of the staff members and orders.

ER MODEL OF GROCERY SHOP



❖ **Landing Page : From here you can Sign up or Login into our grocery app**



Code of Home Page:

React code:

```
import React , {Component} from "react";

import "./homepage.css"

import logo from "./logo.jpg"

import {Link , Button} from "@mui/material";

class Homepage extends React.Component{


  render() {
```

```
    return (
      <div>
        <div className = "banner">
          <div className ="navbar">
            <img src={logo} className={"logo"}/>
              <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">Contact Us</a>
                </li>
              </ul>
          </div>
          <div className="content">
            <h1>GROCERY STORE</h1>
            <p>For the person who loves grocery;</p>
              <p>One stop destination for all your needs</p>
              <p>Just (Name)!</p>
            <div className="links-buttons">
              <Link href = {"/register"}><button id = "signUpButton">SIGN UP<span></span></button></Link>
              <Link href = {"/login"}><button id = "loginButton">LOGIN<span></span></button></Link>
            </div>
```

```
            </div>
          </div>
        </div>
      );
  }
}
export default Homepage
```

CSS code:

```
*{
  margin: 0;
  padding: 0;
  font-family: sans-serif;
}

.banner{
  width: 100%;
  height: 100vh;
  background-image: linear-gradient(rgba(0, 0, 0, 0.75), rgba(0, 0, 0, 0.75)), url(https://res.cloudinary.com/purnesh/image/upload/w_1000,f_auto,q_auto:eco,c_limit/216214122777781.jpg);
  background-size: cover;
```

```
    background-position: center;
}

.navbar{
    width: 85%;
    margin: auto;
    padding: 35px 0;
    display: flex;
    align-items: center;
    justify-content: space-between;
}

.navbar ul li{
    list-style: none;
    display: inline-block;
    margin: 0 20px;
    position: relative;
}

.logo{
    width: 120px;
```

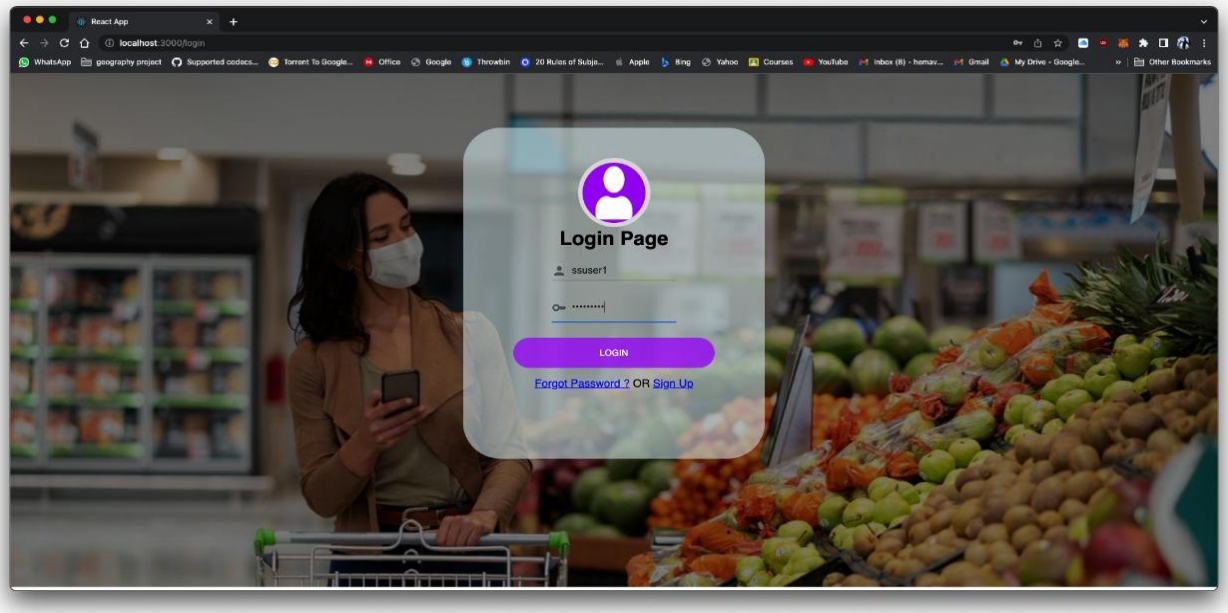
```
    cursor: pointer;
}

.navbar ul li a{
    text-decoration: none;
    color: #fff;
    text-transform: uppercase;
}

.navbar ul li::after{
    content: '';
    height: 3px;
    width: 0;
    background: #009688;
    position: absolute;
    left: 0;
    bottom: -10px;
    transition: 0.5s;
}

.navbar ul li:hover::after{
```

```
    width: 100%;
}

.content{
    width: 100%;
    position: absolute;
    top: 50%;
    transform: translateY(-50%);
    text-align: center;
    color: #fff;
}

.content h1{
    font-size: 70px;
    margin-top: 80px;
}

.content p{
    margin: 20px auto;
    font-size: 20px;
    font-weight: 100;
```

```
    line-height: 25px;
}

#signUpButton , #loginButton{
    width: 200px;
    padding: 15px 0;
    text-align: center;
    margin: 20px 10px;
    border-radius: 25px;
    font-weight: bold;
    border: 2px solid #009688;
    background: transparent;
    color: #fff;
    cursor: pointer;
    position: relative;
    overflow: hidden;
    display: inline;
}

span{
    background: #009688;
```

```
    height: 100%;
    width: 0%;
    border-radius: 25px;
    position: absolute;
    left: 0;
    bottom: 0;
    z-index: -1;
    transition: 0.5s;
}

button#loginButton:hover span , button#signUpButton:hover span{
    width: 200px;
}

button#loginButton:hover , button#signUpButton:hover{
   border: none;
}
```

❖ **Login Page: From here you can login into our grocery app**



Code of Login page:

React code:

```
import './LoginPage.css';

import axios from "axios";

import React from "react";

import a from "./a.png";

import {Button, Input, InputAdornment} from "@mui/material";

import {Cookies} from "react-cookie";

import PersonIcon from '@mui/icons-material/Person';

import KeyIcon from '@mui/icons-material/Key';
```

```
import {Navigate} from "react-router-dom";
class LoginPage extends React.Component {

  initialState = {
    User: {
      username: "",
      email: "",
      full_name: "",
      disabled: false
    },

    jwt: {
      access_token: "",
      token_type: ""
    },

    redirect : false
  }

  state = this.initialState
```

```
render() {
    const bustyle = {
        width: 320,
        height: 50,
        borderRadius: '25px',
        backgroundColor: 'rgba(169,3,252,0.77)',
        color: 'white',
        fontsize: '35px',
        border: 'none'
    }
    const istyle = {
        width: '300',
        height: '50px',
        borderRadius: '60',
        border: 'none',
        outline: 'none',
        backgroundcolor: '#fff'
    }

    if(this.state.redirect){
        return <Navigate to={"/allItems"}/>
```

```
    }

    return (
      <div>
        <div className="main">
          <div className="sub-main">
            <div>
              <div className="imgs">
                <div className="container-image">
                  <img src={a} alt="profile" className="profile"/>
                </div>
              </div>
              <div>
                <h1>Login Page</h1>
                <form id={'login-form'}>
                <div style={{marginTop: '20px'}}>
                  <Input type="text" placeholder={'Enter a username'} startAdornment={
                    <InputAdornment position="start">
                      <PersonIcon></PersonIcon>
                    </InputAdornment>
```

```
                    }
                        name="username" id="username"/>
                    </div>
                    <div className="second-input">
                      <Input type="password" startAdornment={
                        <InputAdornment position="start">
                          <KeyIcon/>
                        </InputAdornment>
                      } placeholder={'Enter password'} style={istyle} name="password" id="password"/>
                    </div>
                    </form>
                    <div className="login-button">
                      <Button value="LOGIN" style={bustyle} id="submitButton"
                          onClick={this.LoginSubmit}> LOGIN </Button>
                    </div>
                    <p className="link" style={{marginTop: '15px'}}>
                      <a href="#">Forgot Password ?</a> OR <a href="/register">Sign Up</a>
                    </p>
                  </div>
                </div>
```

```
            </div>
          </div>
        </div>
      )
  }
  LoginSubmit = async() => {
    const cookie = new Cookies();
      axios.post("http://localhost:8000/token", new FormData(document.getElementById("login-form")))
        .then(res => {
          if (res.status === 200) {
            this.setState({jwt: res.data})
            cookie.set("jwt", res.data.access_token, {maxAge: 60 * 60})
            cookie.set("cart", {})
            return axios.get("http://localhost:8000/users/me", {headers: {"Authorization": `Bearer ${res.data.access_token}`}})
          }
        })
        .then(res => {
          if (res.status === 200) {
            cookie.set("full_name", res.data.full_name, {maxAge: 60 * 60});
            alert("Login Successful");
```

```
            this.setState({User: res.data, redirect: true});
            console.log(res.data);
          }
        })
        .catch(error => {
          if(error.response.status === 401){
          alert("Username or password is Incorrect");
          }

          else if(error.response.status === 422){
            alert("Please enter both username and password")
          }
        })
    }
}
export default LoginPage
```

Code of Home Page:

CSS Code:

```
.main {
```

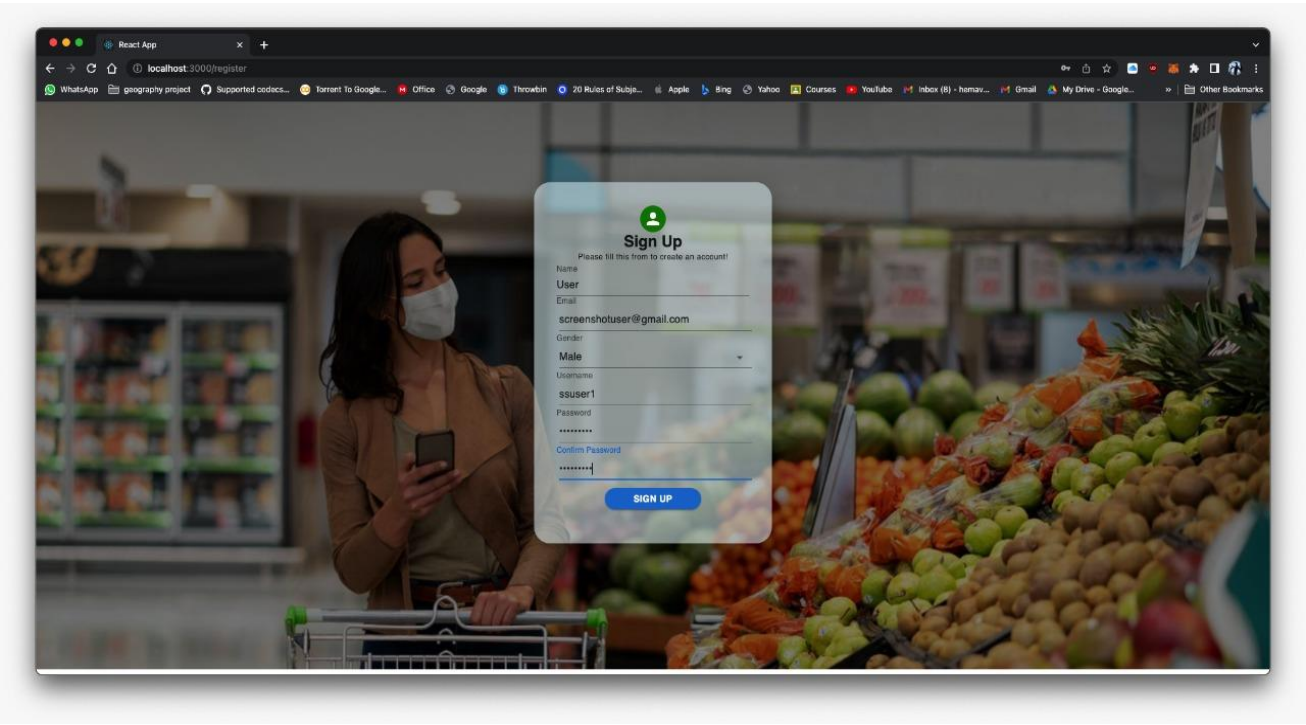
```
    text-align: center;
    justify-content: center;
    display: flex;
    padding-top: 90px;
    padding-bottom: 90px;
    height: 78.3vh;
    background-image: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url('./login.jpg');
    overflow-y: hidden;
    background-size: cover;
    background-repeat: no-repeat;

}

.sub-main {
    display: flex;
    justify-content: center;
    height:520px;
    width: 25%;
    padding-top: 30px;
    border-radius: 60px;
```

```
    background: rgba(217, 235, 238, 0.57);
}
.imgs {
    padding-top: 20px;
    justify-content: center;
    display: flex;

}
.container-image {
    background-color: rgb(223, 221, 221);
    border-radius: 150px;
    align-items: center;
    display: flex;
    justify-content: center;
    height: 115px;
    width: 115px;

}
.profile {
    height: 100px;
    width: 100px;
```

```
    border-radius: 130px;
}
.email{
    height: 25px;
    width: 25px;
    position: absolute;
    padding: 14px 0 0 25px;
}

.name{
    padding-left: 70px;
    font-size: 20px;
    border:1px solid;

}
.second-input{
    padding-top: 20px;
}


.login-button{
```

```
    padding-top: 25px;
}



.link{
    font-size: 18px;
    font-weight: 500;
}
a{
    color: blue;
}
```

❖ **Sign Up Page: The User can sign up here**



Code of Sign-Up page:

React code:

```
import React, { useState } from "react";

import "./Register.css"

import {

  Grid,

  Paper,

  Avatar,

  Typography,

  Button,
```

```
    NativeSelect,
    TextField, InputLabel, FormControl
} from "@mui/material";
import axios from "axios";
import {Navigate, useNavigate} from "react-router-dom";


export const Register=()=> {
    const bstyles = {
        paperContainer: {
            backgroundSize: 'cover',
            backgroundPosition: 'cover',
            width:'100%',
            height:'77vh',
        }
    };
    const navigator = useNavigate()
    const Headerstyle = {marginTop:0}
    const backstyle = {backgroundColor: 'green'}
    const paperstyle = {padding:35,height:'55vh',width:300,margin:"20px auto",borderRadius:'25px',backgroundColor:'rgba(217, 235, 238, 0.57)'};
    function handleSubmit(){
```

```
    let password=document.getElementById("password").value;
    let cPassword=document.getElementById("cnfrm-password").value;
    const form = new FormData()
    form.set("username" , username)
    form.set("email" , email)
    form.set("password" , password)
    form.set("gender" , gender)
    form.set("full_name" , Name)
    let message=document.getElementById("message");
    if(password.length!==0){
      if(password===cPassword){
        message.textContent="";
        axios.post("http://localhost:8000/createUser/" , form).then(res => {if(res.status === 200){alert("Sign Up Successful. Redirecting To Login"); navigator("/login")}})
      }
      else{
        message.textContent="Passwords don't match";
        message.style.backgroundColor="#ff4d4d";
      }
    }
    else{
```

```
      alert("Password can't be empty!");
      message.textContent="";
    }

}

const [confirmPass , setConfirmPass] = useState("")
const [Name , setName] = useState("")
const [email , setEmail] = useState("")
const [password , setPassword] = useState("")
const [gender , setGender] = useState("M")
const [username , setUsername] = useState("")
return (
   <div className={'SignUp'}>
   <Grid  style={bstyles.paperContainer}>
      <Paper elevation={20} style={paperstyle}>
         <Grid align="center">
            <Avatar style={backstyle}>

            </Avatar>
            <h2 style={Headerstyle}>Sign Up</h2>
```

```
            <Typography variant="caption" component="h1">Please fill this from to create an

              account!</Typography>

          </Grid>

          <form>

            <TextField fullWidth label='Name' placeholder="Enter your name" variant='standard' onChange={e => setName(e.target.value)}/>

            <TextField fullWidth label='Email'style={{padding:'4px'}} placeholder="Enter your email" variant='standard' onChange={e => setEmail(e.target.value)}/>

            <FormControl fullWidth style={{padding:'4px'}}>

              <InputLabel variant="standard" htmlFor="uncontrolled-native">

                Gender

              </InputLabel>

              <NativeSelect

                inputProps={{

                  name: 'Gender',

                  id: 'uncontrolled-native',

                }}

                onInput={e => setGender(e.target.value === "Male" ? "M" : "F")}

              >

                <option value={"Male"}>Male</option>
```

```
                <option value={"Female"}>Female</option>

              </NativeSelect>

            </FormControl>

            <TextField fullWidth label='Username' style={{padding:'4px'}}placeholder="Enter your username"

                variant='standard' onChange={e => setUsername(e.target.value)}/>

            <TextField fullWidth id="password" label='Password' style={{padding:'4px'}} type={"password"} placeholder="Enter your password" variant='standard' onChange={e => setPassword(e.target.value)}/>

            <TextField fullWidth id="cnfrm-password" label='Confirm Password'style={{padding:'4px'}} type={"password"} placeholder="Please confirm your password"

                variant='standard' onChange={e => setConfirmPass(e.target.value)}/>

            <p id={"message"}></p>

            <Button variant='contained' style={{padding:'4px',marginTop:'8px',width:'50%',borderRadius:'25px',fontWeight:'600'}} color='primary' onClick={handleSubmit}>Sign Up</Button>

          </form>

        </Paper>

      </Grid>

      </div>

  )
```

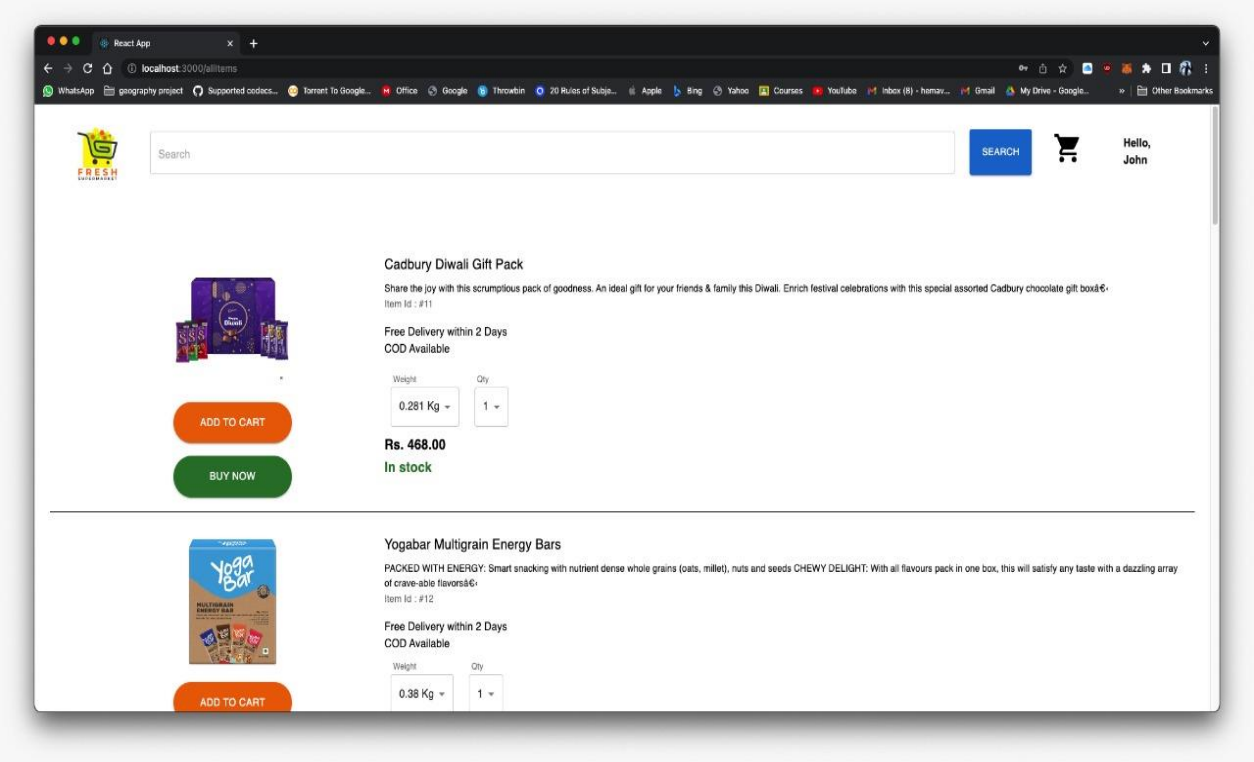
```
}
```

CSS code:

```
.Register{
    text-align: center;
    display: flex;
    min-height: 100vh;
    align-items: center;
    justify-content: center;
    color: rgba(255, 255, 255, 0.59);
}
.SignUp{
    text-align: center;
    justify-content: center;
    display: flex;
    padding-top: 100px;
    padding-bottom: 90px;
    background-image: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)), url('./login.jpg');
    background-size: cover;
    background-repeat: no-repeat;
    height: 77vh;
```

}

❖ **Items Page: Here you can get the view of all items and also search for items you want and also go to the cart page**



Code of All Items page:

React code:

```
import React  from "react";

import axios from "axios";

import {Badge, Button, Grid, TextField, Typography, Link, iconClasses} from "@mui/material";

import logo from "./logo.jpg";

import ParseItems from "./ParseItems";

import {Cookies} from "react-cookie";
```

```
import ShoppingCartIcon from "@mui/icons-material/ShoppingCart";
import {Navigate} from "react-router-dom";

class ProductList extends React.Component{
   initialState = {
      items : [],
      length : 0,
      page_num : 0,
      total_pages : 0,
      redirectToCart : false
   }

   state = this.initialState

   handleSearch = () => {
      let inp = document.getElementById("search-input").value
      if(inp === ""){
         this.componentDidMount()
         return
      }
```

```
    axios.get(`http://localhost:8000/searchItem${inp}`).then(res => this.setState({items : res.data , length : res.data.length , total_pages : Math.ceil(res.data.length / 10) , page_num : 1}))
  }

  userCheck = () => {
    const cookies = new Cookies()
    const name = cookies.get("full_name")
    return !name  ?  "Hello, Guest" : `Hello, ${name.split(" ")[0]}`
  };

  componentDidMount() {
    const cookie = new Cookies()
    const inp = cookie.get("cart-search")
    console.log(inp)
    if (inp !== undefined) {
      axios.get(`http://localhost:8000/searchItem${inp}`).then(res => this.setState({
        items: res.data,
        length: res.data.length,
        total_pages: Math.ceil(res.data.length / 10),
        page_num: 1
```

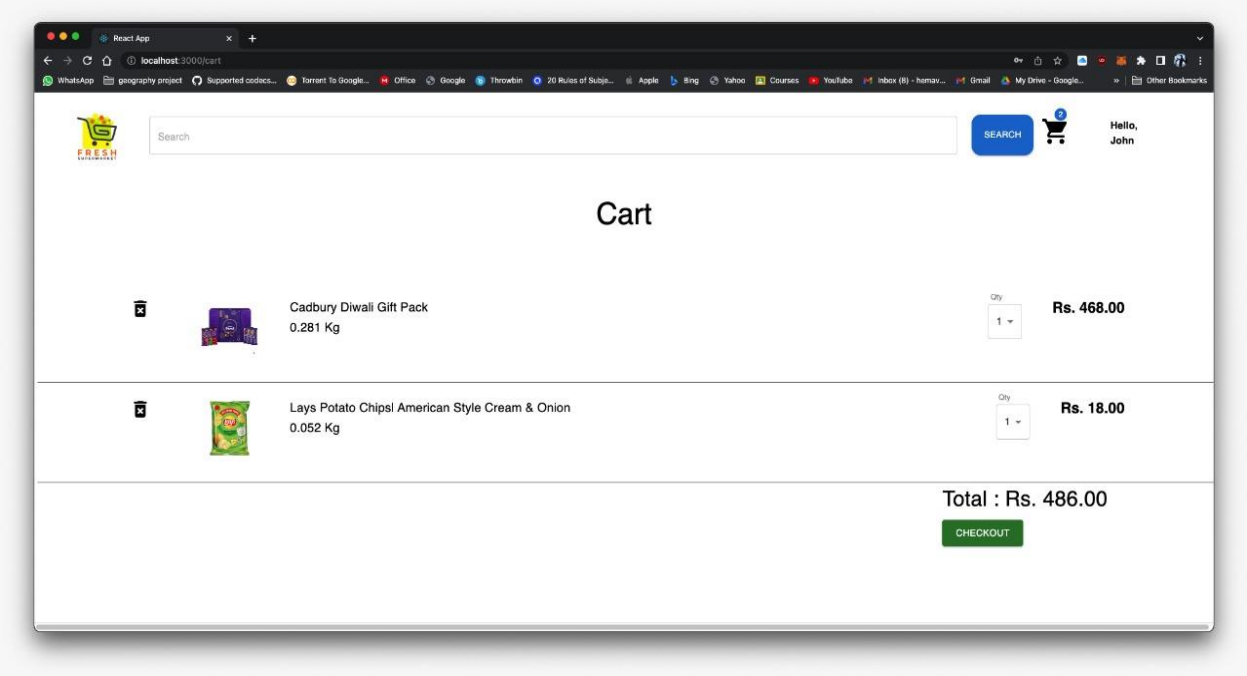
```
        }))
    } else {
        axios.get("http://localhost:8000/getAllItems").then(res => this.setState({
            items: res.data,
            length: res.data.length,
            total_pages: Math.ceil(res.data.length / 10),
            page_num: 1
        }))
        cookie.remove("cart-search")
    }

}

render() {

    if(this.state.redirectToCart){
        return (
            <Navigate to={"/cart"}/>
        )
    }
```

```
if(this.state.items.length === 0){
  return (
    <div>
      <div className ="navbar" style={{width : "93%"}}>
        <Grid container xs spacing={3}>
          <Grid container={true} xs> <img src={logo} className={"logo"}/> </Grid>
          <Grid item xs = {9}> <TextField id = "search-input" fullWidth={true} margin={"dense"} placeholder={"Search"} variant='outlined'/> </Grid>
          <Grid item xs> <Button id = "search-button" onClick={this.handleSearch} variant={"contained"} style={{padding : "20px 20px 20px 20px"}}> Search </Button> </Grid>
          <Grid item xs><ShoppingCartIcon style={{scale : "200%",paddingTop:'10px' , paddingLeft:"15px"}} onClick={() => this.setState({redirectToCart : true})}/></Grid>
          <Grid item xs={0.5}> <Typography paddingTop={1}> <b>{this.userCheck()} </b></Typography> </Grid>
        </Grid>
      </div>
      <Typography color={"error.main"} alignItems={"center"}>No items Found! Please Check your Search Phrase and Try again </Typography>
    </div>
  )
```

```
    }


    return(
        <div>
          <div className ="navbar" style={{width:"93%"}}>
            <Grid container xs spacing={3}>
            <Grid container={true} xs> <img src={logo} className={"logo"}/>
</Grid>
            <Grid item xs = {9}> <TextField id = "search-input" fullWidth={true}
margin={"dense"} placeholder={"Search"} variant='outlined'/> </Grid>
            <Grid item xs> <Button id = "search-button"
onClick={this.handleSearch} variant={"contained"} style={{padding : "20px 20px
20px 20px"}}> Search </Button> </Grid>
              <Grid item xs><ShoppingCartIcon style={{scale :
"200%",paddingTop:'10px' , paddingLeft:"15px"}} onClick={() =>
this.setState({redirectToCart : true})}/></Grid>
            <Grid item xs={0.5}> <Typography paddingTop={1}>
<b>{this.userCheck()} </b></Typography> </Grid>
            </Grid>
          </div>
            <Grid container padding={"20px 20px 20px 20px"}>
              {this.state.items.slice((this.state.page_num - 1) * 10,
this.state.page_num * 10).map(el =>
                <ParseItems id={el.id} item={el}/>)}
```

```
            </Grid>

            <Grid container justifyContent={"center"} padding={"10px 10px 10px 10px"}>

            <Button id={"back-button"} onClick={() => this.setState({page_num : this.state.page_num === 1 ? 1 : this.state.page_num - 1})}> Prev </Button> <span></span> <Button id={"next-button"} onClick={() => this.setState({page_num : this.state.page_num === this.state.total_pages ? this.state.page_num : this.state.page_num + 1})}> Next </Button>

            </Grid>

            <Grid container justifyContent={"center"} padding={"10px 10px 10px 10px"}> {Array.from({length : this.state.total_pages} , (v , k) => k + 1).map(el => <Button id = {el} value={el} onClick={e => this.setState({page_num : e.target.value})}> {el}</Button>)} </Grid>

        </div>

    )

  }

}

export default ProductList
```

❖ **Cart Page: Here you can view all items in cart also delete them and proceed to the payments page**



Code of Cart page:

React code:

```
import React from "react";

import axios from "axios";

import {Cookies} from "react-cookie";

import CartItemParse from "./CartItemParse";

import {Badge, Button, Grid, TextField, Typography} from "@mui/material";

import logo from "./logo.jpg";

import ShoppingCartIcon from '@mui/icons-material/ShoppingCart';
```

```
import {Navigate} from "react-router-dom";
import ProductList from "./ProductList";



class CartView extends React.Component{

   initialState = {
      items : [],
      redirect : false,
      redirectToPayment : false
   }

   state = this.initialState

   componentDidMount() {
      this.getCartItems()
   }

   getCartItems = () =>{
      const token = new Cookies().get("jwt")
```

```
    axios.get("http://localhost:8000/getCart/" + token).then(res => this.setState({items : res.data}))
  }


  handleSearch = () => {
    new Cookies().set("cart-search" , document.getElementById("search-input-cart").value)
    this.setState({redirect : true})
  }
  userCheck = () => {
    const cookies = new Cookies()
    const name = cookies.get("full_name")
    return !name  ?  "Hello, Guest" : `Hello, ${name.split(" ")[0]}`
  };


  getTotal = () => {
    let total = 0;
    this.state.items.map((el) => {total += (el.price * el.qty)});
    return total;
  }
```

```
handleCheckout = () => {
    const cookies = new Cookies()
    cookies.set("amount" , this.getTotal())
    this.setState({redirectToPayment : true})
}

render() {

    if(this.state.redirectToPayment){
        return (
            <Navigate to={"/payment"} state={{amount : this.getTotal()}}/>
        )
    }

    if(this.state.redirect){
        return <Navigate to={"/allItems"}/>
    }

    return(
        <div>
            <div className ="navbar" style={{width:'93%'}}>
```

<Grid container xs spacing={3}>

<Grid container={true} xs> </Grid>

<Grid item xs = {9}> <TextField id = "search-input-cart" fullWidth={true} margin={"dense"} placeholder={"Search"} variant='outlined'/> </Grid>

<Grid item xs> <Button id = "search-button" onClick={this.handleSearch} variant={"contained"} style={{borderRadius:'15px',padding : "20px 20px 20px 20px"}}> Search </Button></Grid>

<Grid item xs><Badge badgeContent={this.state.items.length} color="primary"><ShoppingCartIcon style={{scale : "200%",paddingTop:'10px'}}/></Badge> </Grid>

<Grid item xs> <Typography paddingTop={0.75}> {this.userCheck()} </Typography> </Grid>

</Grid>

<Typography align={"center"} variant={"h3"} paddingBottom={10}>Cart</Typography>

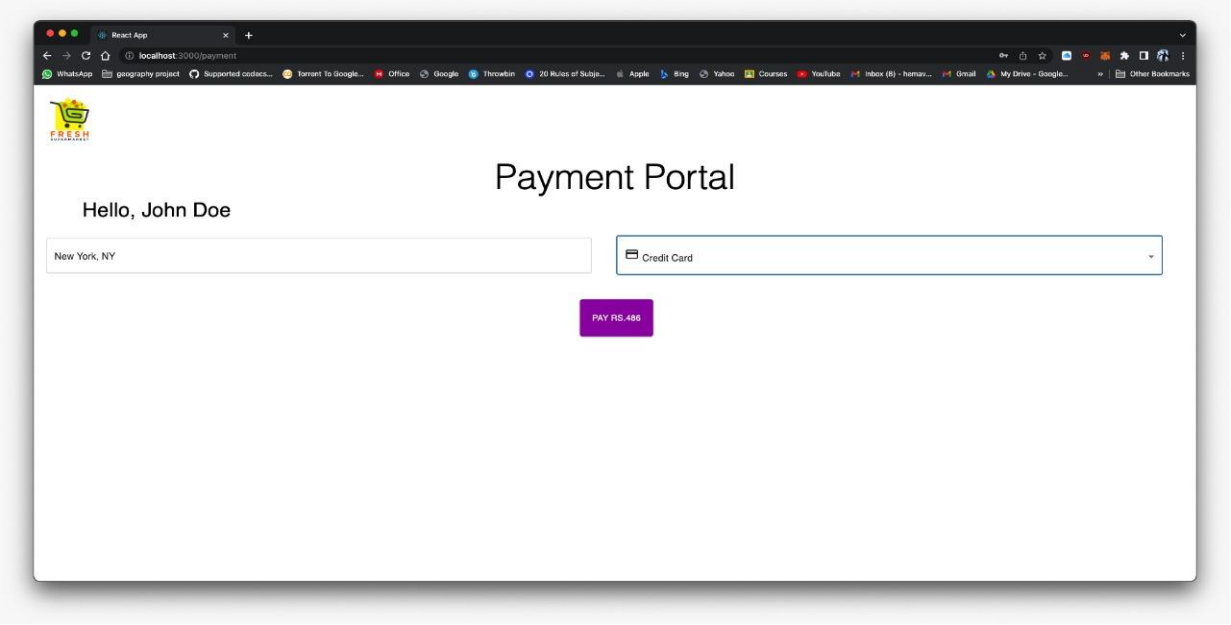
{this.state.items.length === 0 ? <Typography variant={"h4"} color={"error.main"} align={"center"}>No items in Cart </Typography> :

{this.state.items.map(el => <CartItemParse item ={el} parentUpdate = {this.getCartItems}/>)}

<div style={{paddingLeft:"77%"}}>

```
                <b><Typography gutterBottom variant={"h4"}>Total : Rs. {this.getTotal().toFixed(2)}</Typography></b>

                    <Button variant={"contained"} size={"large"} color={"success"} onClick={this.handleCheckout}>Checkout</Button> <span/></div>

                </div>

            }

        </div>

    )

  }

}

export default CartView
```

- **Payments Page: Here you can proceed to make the payments through various options also you need to enter your address here for delivery**



Code of Payments page:

React code:

```
import React from "react";

import logo from "./logo.jpg";

import {Button, MenuItem, TextField, Typography} from "@mui/material";

import {Cookies} from "react-cookie";

import CreditCardIcon from '@mui/icons-material/CreditCard';

import CurrencyRupeeIcon from '@mui/icons-material/CurrencyRupee';

import PaymentsIcon from '@mui/icons-material/Payments';
```

```
import axios from "axios";
import {Navigate} from "react-router-dom";
class Payment extends React.Component{
   initialState = {
      address: "",
      paymentMethod : "Cash On Delivery",
      amount : 0,
      no_items : 0,
      order_id : ""
   }

   state = this.initialState

   handlePayClick = () => {
      const cookie = new Cookies()
      const form = new FormData()
      form.set("user_token" , cookie.get("jwt"))
      form.set("address" , document.getElementById("address").value)
      form.set("payment_method" , this.state.paymentMethod)
      form.set("amount" , cookie.get("amount"))
```

```
    axios.post("http://localhost:8000/checkout" , form).then(res => {if(res.status === 200){this.setState({order_id : res.data}); console.log(res.data); cookie.remove("amount"); alert(`Order #${res.data} placed successfully. Redirecting to items page`)}})

  }



  render() {

    if(this.state.order_id !== ""){

      return(

      <Navigate to={"/allItems"}/>

      )

    }


    return (

      <div>

        <img src={logo} className={"logo"}/>

        <Typography variant={"h2"} align={"center"}>Payment Portal</Typography>

        <Typography variant={"h4"} gutterBottom paddingLeft={10}> Hello, {new Cookies().get("full_name")}</Typography>

        <TextField style={{width : "47%" , padding : "20px 20px 20px 20px"}} id={'address'} placeholder={"Enter Address"} variant={"outlined"}></TextField>
```

```
        <TextField select style={{width : "47%" , padding : "15px 10px 10px 20px"}}
          labelId="payment-selector"
          id="payment-method"
          defaultValue={"COD"}
          onChange={e => this.setState({paymentMethod : e.target.value})}
        >
          <MenuItem value={"Credit Card"} >{<CreditCardIcon/>} Credit Card</MenuItem>
          <MenuItem value={"Debit Card"}>{<CreditCardIcon/>} Debit Card</MenuItem>
          <MenuItem value={"UPI"}>{<CurrencyRupeeIcon/>} UPI</MenuItem>
          <MenuItem value={"COD"}>{<PaymentsIcon/>} Cash On Delivery</MenuItem>
        </TextField> <br/>
        <div style={{paddingLeft : "47%" , paddingTop : "25px"}}>
        <Button variant={"contained"} style={{padding : "20px 20px 20px 20px"}} color={"secondary"} onClick={this.handlePayClick}>Pay Rs.{new Cookies().get("amount")}</Button>
        </div>
      </div>
    );
  }
```

```
}

export default Payment
```

❖ Main Backend Code

```
import asyncio

from datetime import datetime, timedelta

from typing import Optional

from fastapi import Depends, FastAPI, HTTPException, status, Form

from fastapi.security import OAuth2PasswordBearer, OAuth2PasswordRequestForm

from jose import JWTError, jwt

from passlib.context import CryptContext

from pydantic import BaseModel

from starlette.middleware.cors import CORSMiddleware

import mysql.connector



db = mysql.connector.connect(user = "root" , password = "root" , host = "localhost");

cur = db.cursor(buffered=True)

cur.execute("USE grocery_shop_management;")


app = FastAPI(debug = True)
```

```
app.add_middleware(CORSMiddleware , allow_origins = ["http://localhost:3000"] , allow_credentials = True , allow_headers = ['*'] , allow_methods = ['*'])

SECRET_KEY = "efd1a9ccdb325278a5b2d8183d3bf005a17bab75609ff4fc90e83f75ef9ec617"

ALGORITHM = "HS256"

ACCESS_TOKEN_EXPIRE_MINUTES = 60

CHECK = True



class Token(BaseModel):

    access_token: str

    token_type: str



class TokenData(BaseModel):

    username: str | None = None



class User(BaseModel):

    username: str

    email: str | None = None
```

```
    full_name: str | None = None
    disabled: bool | None = None

class UserForm(BaseModel):
    username: str
    email: str | None = None
    full_name: str | None = None
    password : str
    gender : str


class UserInDB(User):
    hashed_password: str

class GroceryItem(BaseModel):
    id : int
    name : str
    desc : str
    image_link : str
    type : int
    details : list
```

```
class CartItem(BaseModel):
    id : int
    name : str
    image_link : str
    type : int
    type_qty : float
    qty : float
    price : float
    stock_available : int


class OrderedItems(BaseModel):
    item_id : int
    weight : float | None = None
    volume : float | None = None
    price : float
    qty : int
class PreviousOrders(BaseModel):
    id : int
    time : datetime
    address : str
```

```
    amount : float
    payment_type : str
    items : list


pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")


def verify_password(plain_password, hashed_password):
    return pwd_context.verify(plain_password, hashed_password)


def get_password_hash(password):
    return pwd_context.hash(password)


def get_user(username: str):
    cur.execute(f"SELECT * FROM Customer WHERE Username LIKE '{username}'")
```

```
    res = cur.fetchone()
    if(res is None):
        return res
    return UserInDB(username = res[3] , email = res[2] , full_name = res[0] , disabled = False , hashed_password = res[-2])


def authenticate_user(username: str, password: str):
    user = get_user(username)
    if user is None:
        return False
    if not verify_password(password, user.hashed_password):
        return False
    return user


def create_access_token(data: dict, expires_delta: timedelta | None = None):
    to_encode = data.copy()
    if expires_delta:
        expire = datetime.utcnow() + expires_delta
    else:
```

```
    expire = datetime.utcnow() + timedelta(minutes=15)
  to_encode.update({"exp": expire})
  encoded_jwt = jwt.encode(to_encode, SECRET_KEY, algorithm=ALGORITHM)
  return encoded_jwt



async def get_current_user(token: str = Depends(oauth2_scheme)):
  credentials_exception = HTTPException(
    status_code=status.HTTP_401_UNAUTHORIZED,
    detail="Could not validate credentials",
    headers={"WWW-Authenticate": "Bearer"},
  )
  try:
    payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
    username: str = payload.get("sub")
    if username is None:
      raise credentials_exception
    token_data = TokenData(username=username)
  except JWTError:
    raise credentials_exception
```

```
    user = get_user(username=token_data.username)
    if user is None:
        raise credentials_exception
    return user


async def get_current_active_user(current_user: User = Depends(get_current_user)):
    if current_user.disabled:
        raise HTTPException(status_code=400, detail="Inactive user")
    return current_user


def check_username_unique(username):
    cur.execute(f"SELECT * FROM Customer WHERE Username LIKE '{username}';")
    return cur.fetchone() is None


def create_cust_in_db(name , gender , email_id , username , password ):
    hashed = get_password_hash(password)
```

```
    cur.execute(f"INSERT INTO Customer(Name , Gender , Email_ID , Username , Hashed_pass) VALUES('{name}' , '{gender}' , '{email_id}' , '{username}' ,'{hashed}')")

    db.commit()



async def cancel_reservations():

    cur.execute("SELECT * FROM item_reservation WHERE TIMESTAMPDIFF(MINUTE , time_reserved , CURRENT_TIMESTAMP) > 15;")

    det = cur.fetchall()

    for i in det:

        cur.execute(f"SELECT type FROM Items WHERE Item_ID = {i[1]}")

        t = cur.fetchone()[0]

        if t == 1:

            cur.execute(f"UPDATE items_weight SET Stock = Stock + {i[3]} WHERE Item_ID = {i[1]} AND Weight = {i[2]};")

        else:

            cur.execute(f"UPDATE items_volume SET Stock = Stock + {i[3]} WHERE Item_ID = {i[1]} AND Volume = {i[2]};")

        cur.execute(f"DELETE FROM item_reservation WHERE Cust_ID = {i[0]};")

    db.commit()

async def cron_job_cancel_reservations():

    while CHECK:
```

```
        await asyncio.gather(
            asyncio.sleep(100),
            cancel_reservations()
        )



@app.on_event("startup")
async def start_cron():
    asyncio.create_task(cron_job_cancel_reservations())


@app.on_event("shutdown")
def shutdown():
    CHECK = False


@app.post("/token", response_model=Token)
async def login_for_access_token(form_data: OAuth2PasswordRequestForm =
Depends()):
    user = authenticate_user(form_data.username, form_data.password)
    if not user:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
```

```
        detail="Incorrect username or password",
        headers={"WWW-Authenticate": "Bearer"},
    )
  access_token_expires = timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES)
  access_token = create_access_token(
    data={"sub": user.username}, expires_delta=access_token_expires
  )
  return {"access_token": access_token, "token_type": "bearer"}


@app.get("/users/me/", response_model=User)
async def read_users_me(current_user: User = Depends(get_current_active_user)):
  return current_user


@app.get("/users/me/items/")
async def read_own_items(current_user: User = Depends(get_current_active_user)):
  return [{"item_id": "Foo", "owner": current_user.username}]
```

```
@app.post("/createUser/")
async def createUser(username : str = Form(...) , full_name : str = Form(...) , gender : str = Form(...) , email : str = Form(...) , password : str = Form(...)):
    if not check_username_unique(username):
        raise HTTPException(
            status_code = status.HTTP_409_CONFLICT,
            detail = "Username already exists"
        )
    create_cust_in_db(full_name , gender , email , username , password)
    return {"Status" : "Success"}



@app.get("/getAllItems" , response_model=list[GroceryItem])
def getAllItems():
    cur.execute("SELECT * FROM Items;")
    items = cur.fetchall()
    Grocery_items = []
    table_name = ""
    for i in items:
        details = []
        if i[3] == 1:
```

```
        table_name = "items_weight"
    elif i[3] == 2:
        table_name = "items_volume"
    cur.execute(f"SELECT * FROM {table_name} WHERE Item_ID = {i[0]}")
    det = cur.fetchall()
    for j in det:
        details.append([j[1] , j[2] , j[3]])
    if i[3] == 1:
        Grocery_items.append(GroceryItem(id = j[0] , name = i[1] , desc = i[2] , image_link = i[-1] , type = 1 , details = details))
    elif i[3] == 2:
        Grocery_items.append(GroceryItem(id = j[0] , name = i[1] , desc = i[2] , image_link = i[-1] , type = 2 , details = details))

  return Grocery_items

@app.get("/getItem/{id}{qty}" , response_model=CartItem)
def getItemByID(id , qty):
  cur.execute(f"SELECT * FROM Items WHERE Item_ID = {id};")
  i = cur.fetchone()
  table_name = ""
  type = ""
```

```
    if i is None:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail = "Item not found"
        )
    if i[3] == 1:
        table_name = "items_weight"
        type = "Weight"
    elif i[3] == 2:
        table_name = "items_volume"
        type = "Volume"
    cur.execute(f"SELECT Price FROM {table_name} WHERE Item_ID = {i[0]} AND {type} = {qty};")
    det = cur.fetchone()
    print(det)
    return CartItem(id = i[0] , name=i[1], desc=i[2], image_link=i[-1], type=i[3], qty = qty , price = det[0])


@app.get("/searchItem{word}")
def getSearchResults(word):
    cur.execute(f"SELECT * FROM Items WHERE Name LIKE '%{word}%';")
    items = cur.fetchall()
```

```
    Grocery_items = []
    table_name = ""
    for i in items:
        details = []
        if i[3] == 1:
            table_name = "items_weight"
        elif i[3] == 2:
            table_name = "items_volume"
        cur.execute(f"SELECT * FROM {table_name} WHERE Item_ID = {i[0]}")
        det = cur.fetchall()
        for j in det:
            details.append([j[1], j[2], j[3]])
        if i[3] == 1:
            Grocery_items.append(GroceryItem(id=j[0], name=i[1], desc=i[2], image_link=i[-1], type=1, details=details))
        elif i[3] == 2:
            Grocery_items.append(GroceryItem(id=j[0], name=i[1], desc=i[2], image_link=i[-1], type=2, details=details))
    return Grocery_items


@app.post("/reserveItem")
```

```
def reserveItem(item_id : int = Form(...) , item_type : float = Form(...)  , item_qty : int = Form(...)  , token : str = Form(...)  , type : int = Form(...)):

    payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])

    username = payload.get("sub")

    cur.execute(f"SELECT Cust_ID FROM Customer WHERE Username LIKE '{username}';")

    user_id = cur.fetchone()[0]

    cur.execute(f"SELECT * FROM item_reservation WHERE Cust_ID = {user_id} AND Item_ID = {item_id} AND Item_type = {item_type}")

    res = cur.fetchone()

    old_qty = 0

    if res is not None:

        old_qty = res[3]

        cur.execute(f"UPDATE item_reservation SET Item_qty = {item_qty} WHERE Item_ID = {item_id} AND Item_type = {item_type};")

    else:

        cur.execute(f"INSERT INTO item_reservation(Cust_ID, Item_ID, Item_type, Item_qty) VALUES ({user_id} , {item_id} , {item_type} , {item_qty});")

    if type == 1:

        cur.execute(f"UPDATE items_weight SET Stock = Stock - {item_qty - old_qty} WHERE Item_ID = {item_id} AND Weight = {item_type};")

    else:

        cur.execute(f"UPDATE items_volume SET Stock = Stock - {item_qty - old_qty} WHERE Item_ID = {item_id} AND Volume = {item_type};")
```

```
    db.commit()



@app.get("/getCart/{token}")
def getCartItems(token : str):
    payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
    username = payload.get("sub")
    cur.execute(f"SELECT Cust_ID FROM Customer WHERE Username LIKE '{username}';")
    user_id = cur.fetchone()[0]
    cur.execute(f"SELECT * FROM item_reservation WHERE Cust_ID = {user_id}")
    det = cur.fetchall()
    items = []
    for i in det:
        cur.execute(f"SELECT name , image_link , type FROM Items WHERE Item_ID = {i[1]}")
        item_details = cur.fetchone()
        item_price_stock = []
        if item_details[2] == 1:
            cur.execute(f"SELECT Price , Stock FROM items_weight WHERE Item_ID = {i[1]} AND Weight = {i[2]}")
            item_price_stock = cur.fetchone()
```

```
    else:

        cur.execute(f"SELECT Price , Stock FROM items_volume WHERE Item_ID = {i[1]} AND Volume = {i[2]}")

        item_price_stock = cur.fetchone()

    items.append(CartItem(id = i[1] , name = item_details[0] , image_link = item_details[1] , type = item_details[2] , type_qty = i[2] , qty = i[3] , price = item_price_stock[0] , stock_available = item_price_stock[1]))

  return items


@app.post("/deleteFromCart")

def deleteItemFromCart(item_id : int = Form(...) , item_type : float = Form(...)  , item_qty : int = Form(...)  , token : str = Form(...)  , type : int = Form(...)):

  payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])

  username = payload.get("sub")

  cur.execute(f"SELECT Cust_ID FROM Customer WHERE Username LIKE '{username}';")

  user_id = cur.fetchone()[0]

  cur.execute(f"DELETE FROM item_reservation WHERE Cust_ID = {user_id} AND Item_ID = {item_id} AND Item_type = {item_type};")

  if type == 1:

    cur.execute(f"UPDATE items_weight SET Stock = Stock + {item_qty} WHERE Item_ID = {item_id} AND Weight = {item_type}")

  else:
```

```
    cur.execute(f"UPDATE items_volume SET Stock = Stock + {item_qty} WHERE Item_ID = {item_id} AND Volume = {item_type}")

    db.commit()


@app.post("/checkout")

def checkOut(user_token : str = Form(...) , address : str = Form(...) , payment_method : str = Form(...) , amount : float = Form(...)):

    payload = jwt.decode(user_token, SECRET_KEY, algorithms=[ALGORITHM])

    username = payload.get("sub")

    cur.execute(f"SELECT Cust_ID FROM Customer WHERE Username LIKE '{username}';")

    user_id = cur.fetchone()[0]

    cur.execute(f"INSERT INTO Orders(Amount, Cust_ID, Payment_type , Address) VALUES ({amount} , {user_id} , '{payment_method}' , '{address}');")

    cur.execute(f"SELECT Order_ID FROM Orders WHERE Cust_ID = {user_id} ORDER BY Order_date_time DESC")

    order_id = cur.fetchall()[0][0]

    cur.execute(f"SELECT * FROM item_reservation WHERE Cust_ID = {user_id}")

    det = cur.fetchall()

    items = []

    for i in det:

        cur.execute(f"SELECT type FROM Items WHERE Item_ID = {i[1]}")

        type = cur.fetchone()[0]
```

```
        item_price = 0

        if type == 1:

            cur.execute(f"SELECT Price FROM items_weight WHERE Item_ID = {i[1]} AND Weight = {i[2]}")

            item_price = cur.fetchone()[0]

        else:

            cur.execute(f"SELECT Price FROM items_volume WHERE Item_ID = {i[1]} AND Volume = {i[2]}")

            item_price = cur.fetchone()[0]

        cur.execute(f"INSERT INTO orders_items(Order_Id, Item_Id, Item_type, Type, Item_price, Qty) VALUES ({order_id} , {i[1]} , {i[2]} , {type} , {item_price} , {i[3]})")

    cur.execute(f"DELETE FROM item_reservation WHERE Cust_ID = {user_id}")

    db.commit()

    return order_id


@app.get("/previousOrders/{token}")

def getAllOldOrders(token : str):

    payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])

    username = payload.get("sub")

    cur.execute(f"SELECT Cust_ID FROM Customer WHERE Username LIKE '{username}';")

    user_id = cur.fetchone()[0]
```

```
    orders = []
    cur.execute(f"SELECT * FROM Orders WHERE Cust_ID = {user_id}")
    det = cur.fetchall()
    for i in det:
        cur.execute(f"SELECT * FROM orders_items WHERE Order_Id = {i[0]};")
        items = cur.fetchall()
        items_in_order = []
        for item in items:
            if(item[3] == 1):
                items_in_order.append(OrderedItems(item_id = item[1] , price = item[4]
, qty = item[5] , weight = item[2]))
            else:
                items_in_order.append(OrderedItems(item_id = item[1] , price = item[4]
, qty = item[5] , volume = item[2]))
        orders.append(PreviousOrders(id = i[0] , amount = i[1] , payment_type = i[3]
, time = i[4] , address = i[5] , items = items_in_order))


    return orders




if __name__ == "__main__":
```

```
import uvicorn

uvicorn.run(app)
```

CONCLUSION:

Hereby we implemented a grocery shop app with various functionalities as described above.