JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

# Project Synopsis

## Online Pizza Ordering System

SEMESTER : 6th SEMESTER

BRANCH : INFORMATION TECHNOLOGY

BATCH : B-11

COURSE CODE : 20B12CS315

COURSE NAME: Web Technology and Cyber Security

STUDENT DETAILS: Patil Amit Gurusidhappa - 19104004

Sanjoli Goyal- 19104007

# Table Of Contents

| | |
|---|---|
| **Group Number** | 3 |
| **Project Title** | Online Pizza Ordering System |
| **File Name** | 3_WT.zip |
| **Faculty** | Dr. P. Raghu Vamsi |

**Group Members and Contribution Details**

| Sno | Roll Number | Name | E-mail | Contribution in this work (write your contribution in this work such as task done, tools explored, knowledge gained and presented, etc.) |
|---|---|---|---|---|
| 1 | 19104004 | Patil Amit Gurusidhappa | 19104004@mail.jiit.ac.in | Screen Built HomeScreen, Login Screen, Registration Screen<br><br>Node js Routing order Routes, Pizza Routes and user routes<br><br>**Tools:**<br>Mocha,chai,node.js,express,supertest,nodemon<br>**Databases:**<br>MongoDb,mongoDb<br>Payment Integration: Stripe<br><br>Project synopsis |

| | | | | |
|---|---|---|---|---|
| 2 | 19104007 | Sanjoli Goyal | 19104007@mail.jiit.ac.in | Screen Built HomeScreen, Admin Panel, Cartscreen, orderlist screen Filter Feature, Loading Screen<br><br>Tools used bootstrap, dotenv, react, react-bootstrap, react-dom<br><br>Component Built like Navbar, Error and success, Filter<br><br>Project report |

**Declaration**

I/We hereby declare that this submission is my/our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text. I/We accept the use of the material presented in this report for Education/Research/Teaching purpose by the faculty.

| | |
|---|---|
| Signature<br>  | Signature<br>  |
| Name<br>Sanjoli Goyal<br><br>Date & Place<br>Noida<br>11 april 2022 | Name<br>Patil Amit Gurusidhappa<br>**Date & Place**<br>Noida<br>11 april 2022 |

**Abstract :**

The "Pizza Ordering System" has been developed to override the problems prevailing in the participating manual system. This software is supported to eliminate and in some cases reduce the hardships faced by the existing system. Moreover this system is designed for the particular need of the company to carry out operations in a smooth and effective manner. The application is reduced as much as possible to avoid errors while entering the data. No formal knowledge is needed for the user to use this system.

The main objective of the Pizza Ordering System is to manage the details of Payments, Customer, Coupons,Pizza, Order Status. It manages all the information about Payments, Online Order, Order Status, Payments. The project is totally built at the administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Payments, Customer, Online Order, Coupons. It tracks all the details about the Coupons,Pizza, Order Status. The purpose of Pizza Ordering System is to automate the existing manual system by the help of computerized equipment and full-fledged computer software,fulfilling their equipment, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

**Scope of the project**

**Functional requirements :**

1. **Admin:**

    a. **Add Pizza:**

    Add different types of pizza's in the veg and non-veg category.

    b. **View/Edit/Delete:**

    Can view/update/delete the added pizza from the database.

    c. **View/Update Order:**

    Can view all the orders received from the customer and change the order details accordingly.

d. **Orders List :**

Can view Ordered details.

2. **User:**

**A. Registration:**

Users can register his details.

**B. Login:**

User Login his account.

**C. Home page:**

Users can visit his home page.

**D. Pizza Detail:**

Can view pizza details by selecting a pizza and view its details such as price, toppings, etc…

**E. Add to Cart:**

Customers can add the selected pizza into cart and can check out further.

**F. Buy Now**:

Can buy a selected pizza and can also enter required toppings (If needed) and specify the quantity.

**G. Change Password:**

User can change his current password and make a new password.

## Tools and technologies used:

1. **Frontend** - React.js, Html, Javascript,CSS
2. **Backend** - Node.js Express

3. **Hosting** - Heroku
4. **Database** - Mongoose & MongoDB

## Non functional requirements

### 1. Portability

Applications could be accessed by mobile or desktop devices.

### 2. Security

Email password authentication is being used as well as database security using tokens will be maintained

### 3. Scalability

Scaling of the database is being handle by mongoDB and hosting of the website is being done on the heroku

### 4. Reusability

DRY(Do not repeat yourself) principle is the main thought kept while developing the application and multiple component based applications architecture is used

### 5. Flexibility

This project can be a stepping up project for further advances and always leaves a room for improvement and extension of functionalities.

# Description of the modules of the project.

### 1. Client

/public/index.html - frontend html entry point of the application
/src - Pizza app website components and react state logic is mentioned here

### 2. Models

Contains three models of orderModel PizzaModel and useModel

### 3. Routes

Backend Node.js routes namely for OrderRouts, PizzaRouts and userRouts

### 4. server.js

Starting point of nodejs application

### 5. Db.js

Connection string and logic for connecting to mongoDB is mentioned here

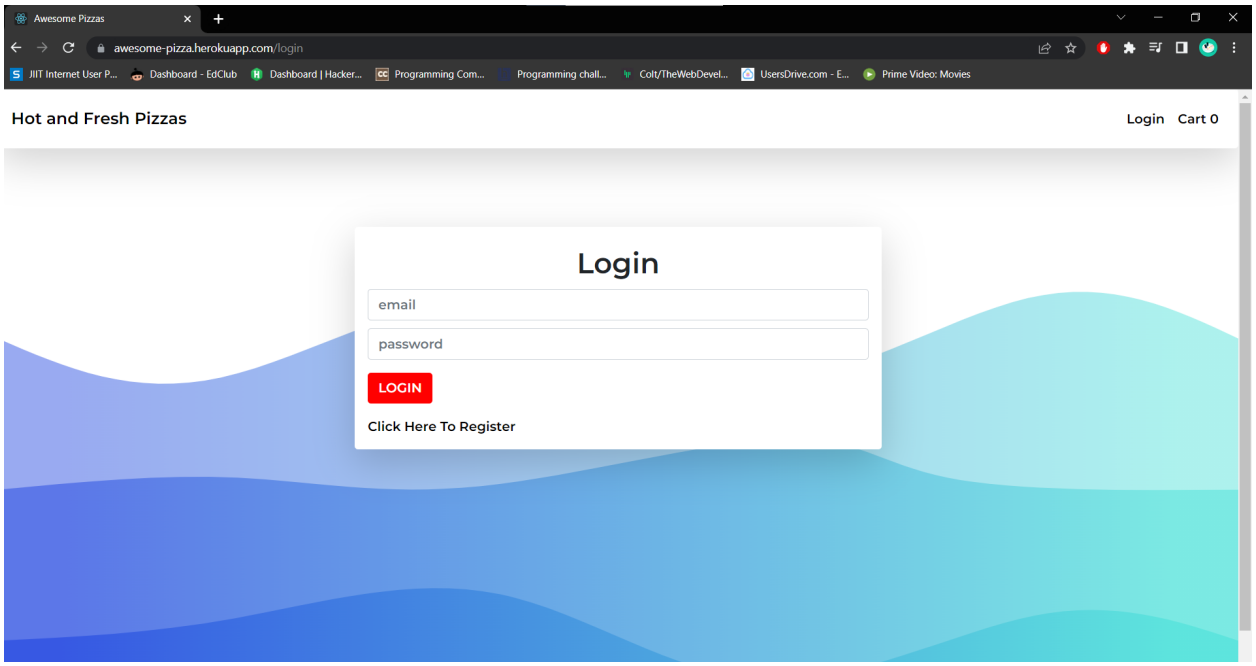
### 6. package .json

Contains script and npm modules are mentioned here

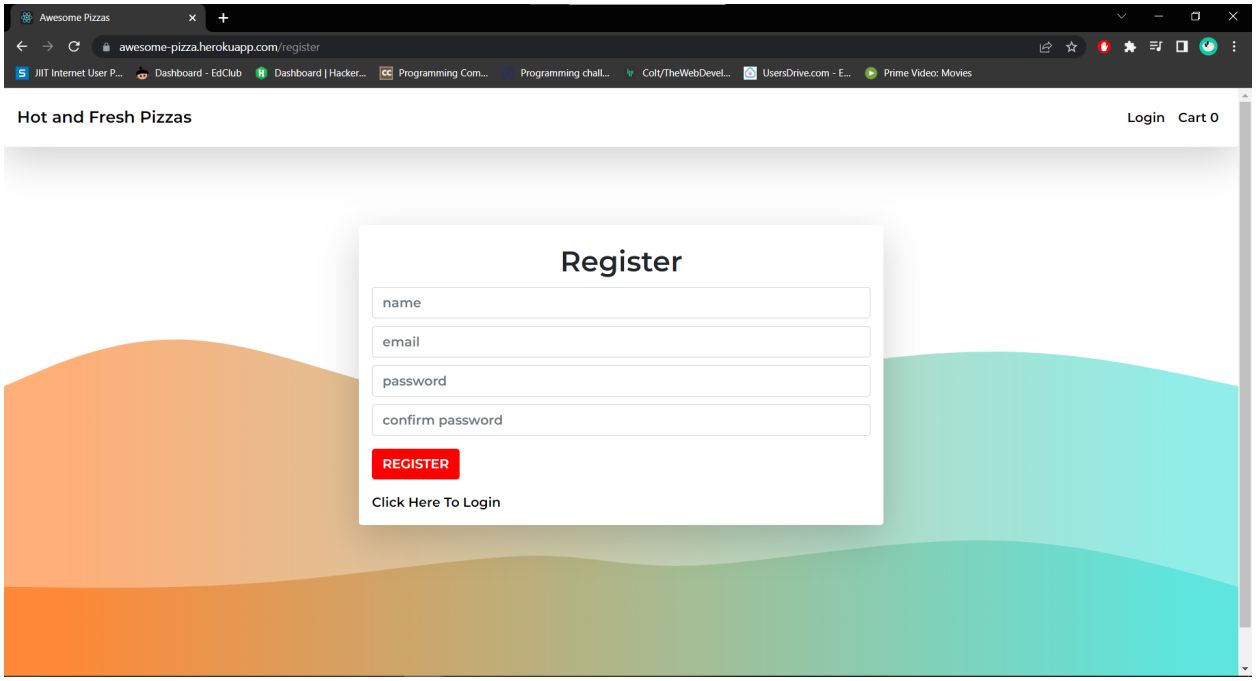
## Design of the project

## Users:

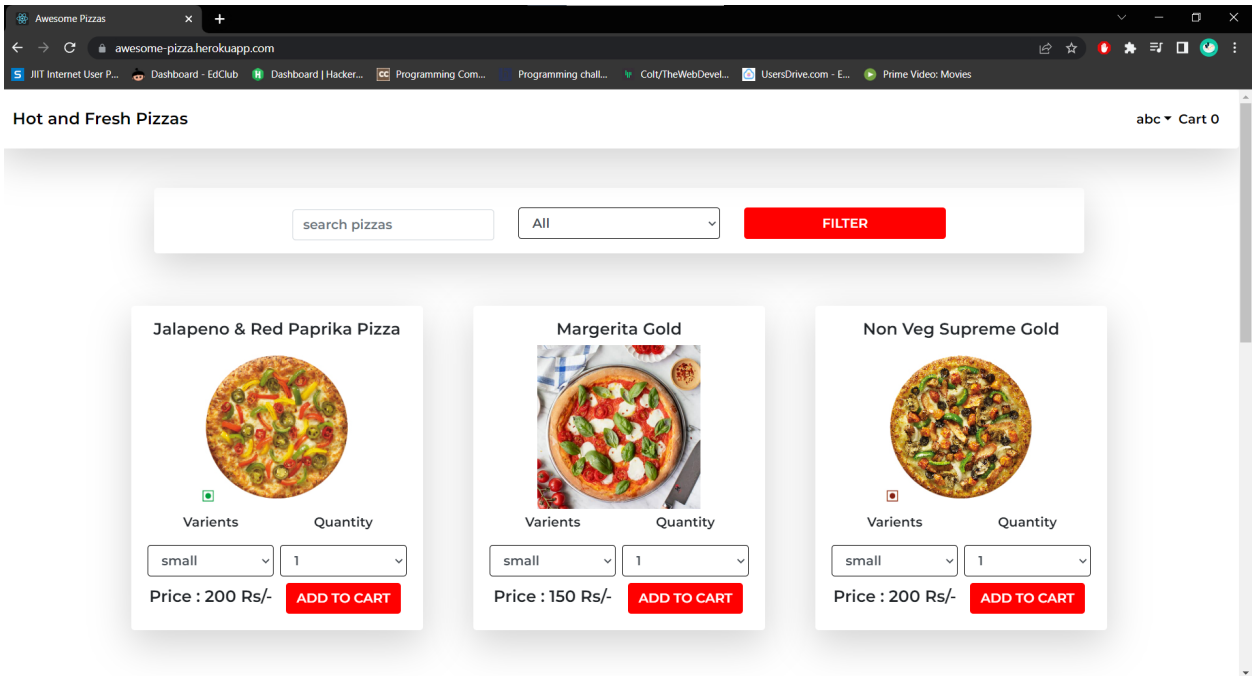
### Login Screen:



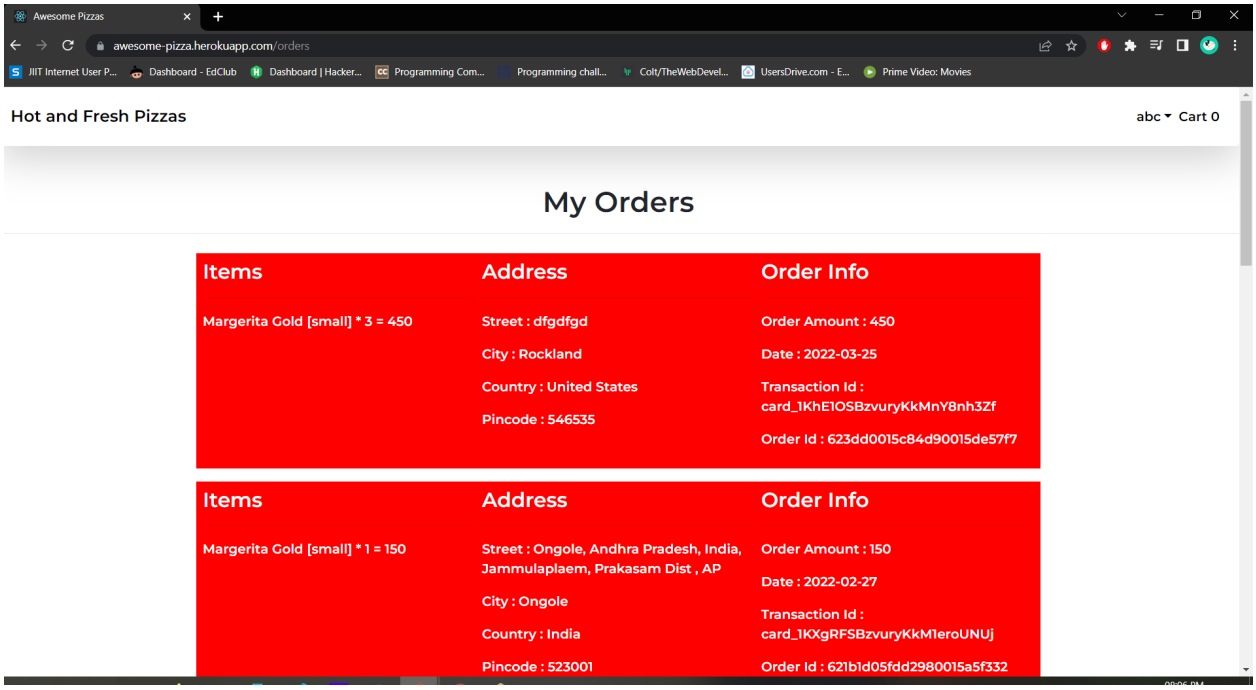
## Register Screen:



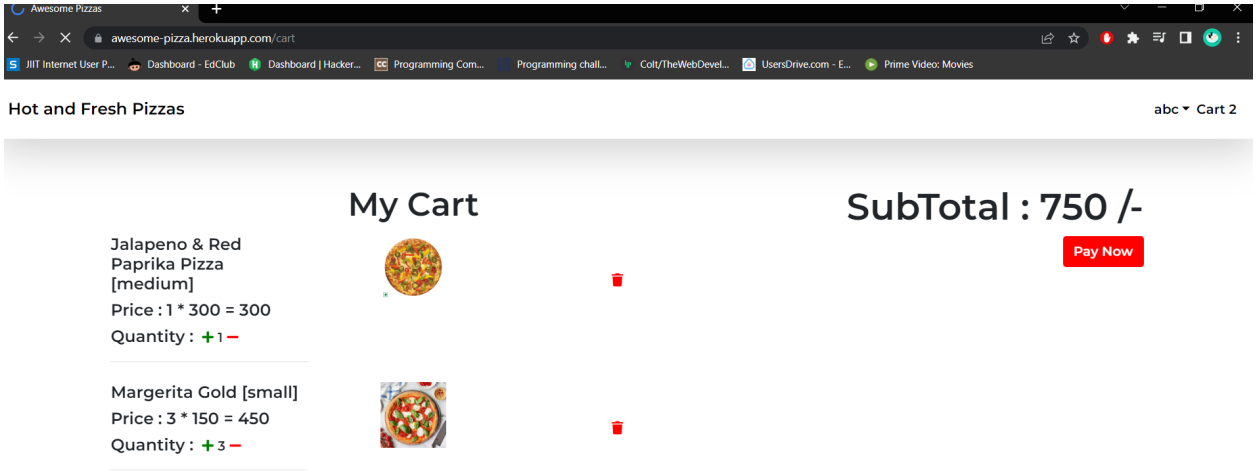

## Home Screen:



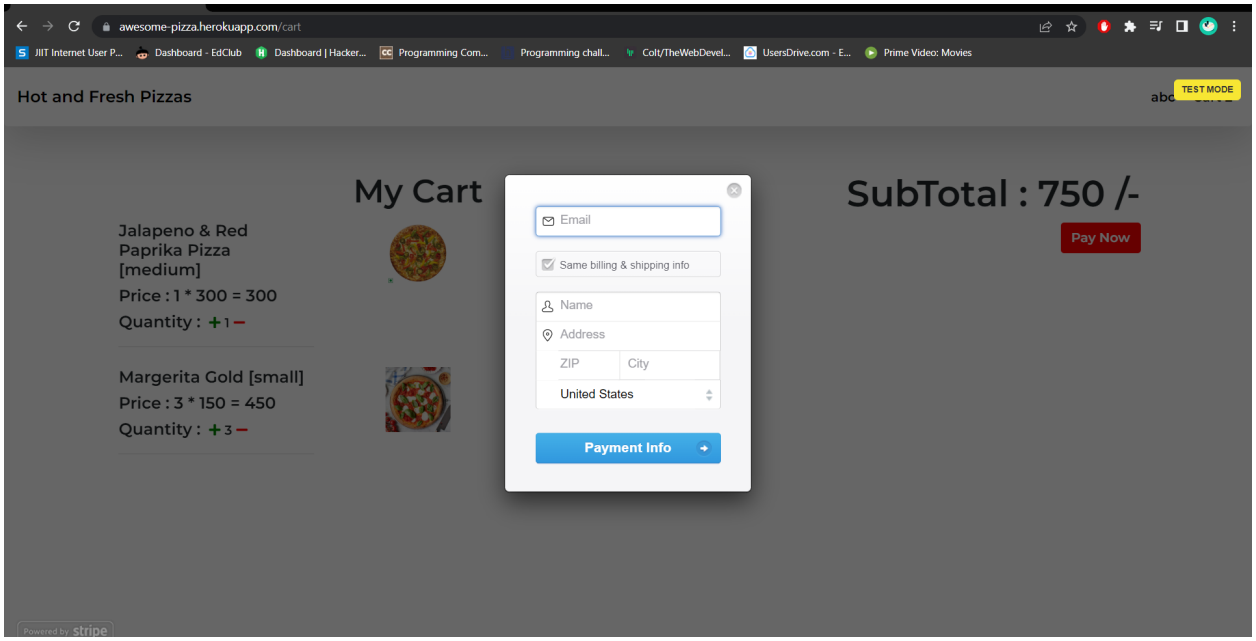
## My Orders Screen:




## Add To Cart Screen:

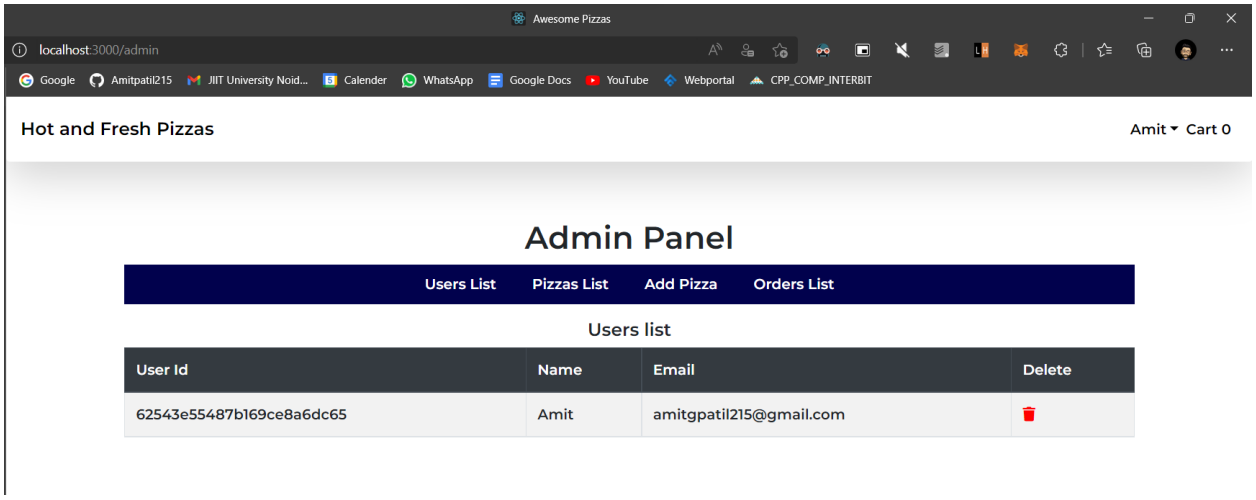


## Payment Screen:

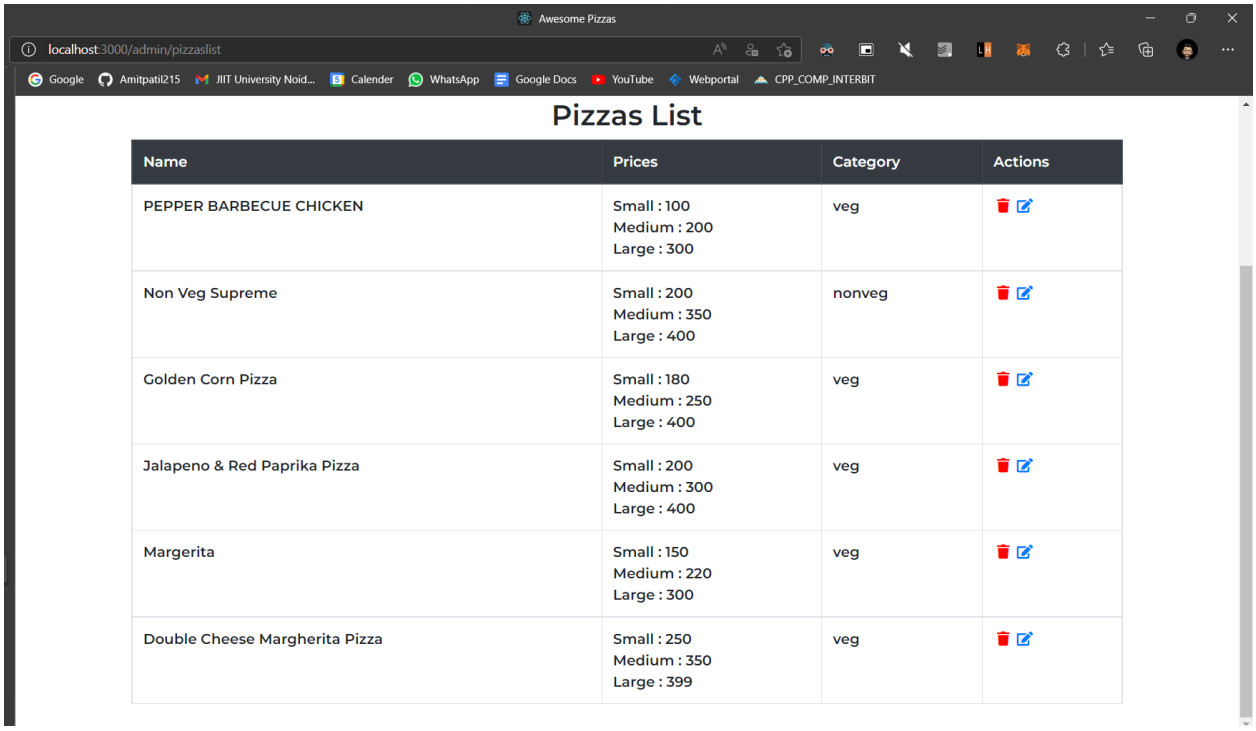



## Admin:

Admin Panel

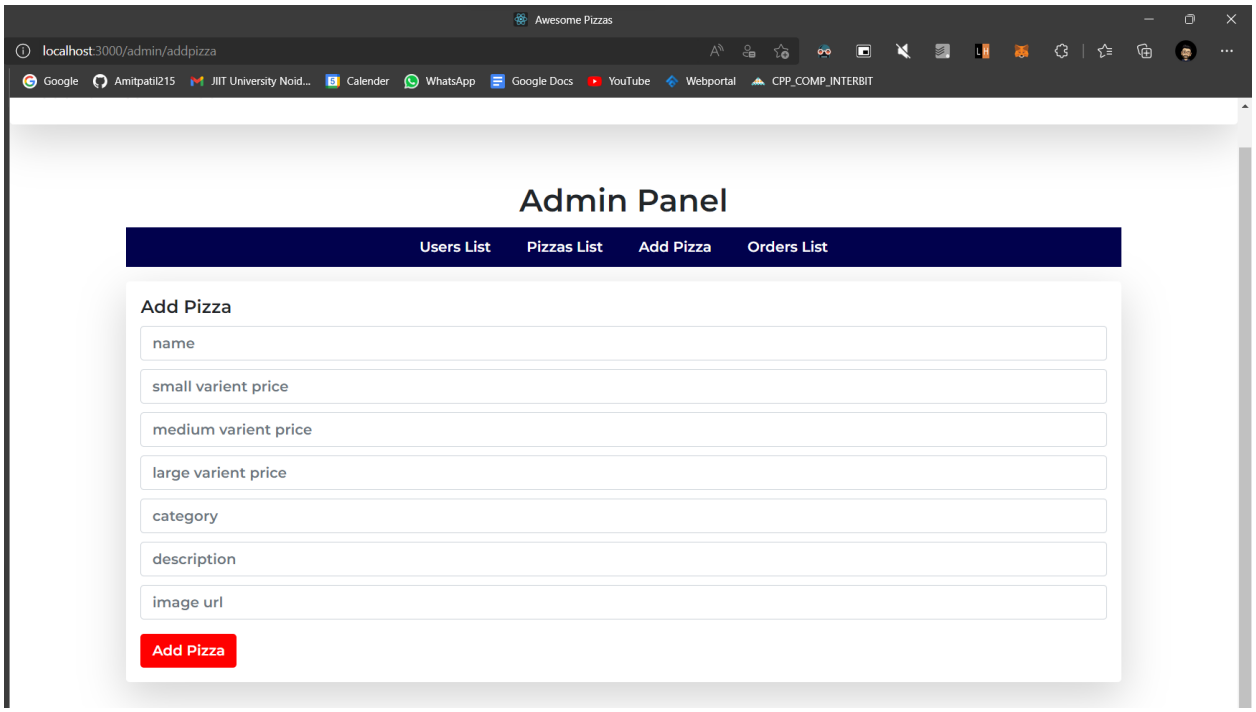



Pizza List

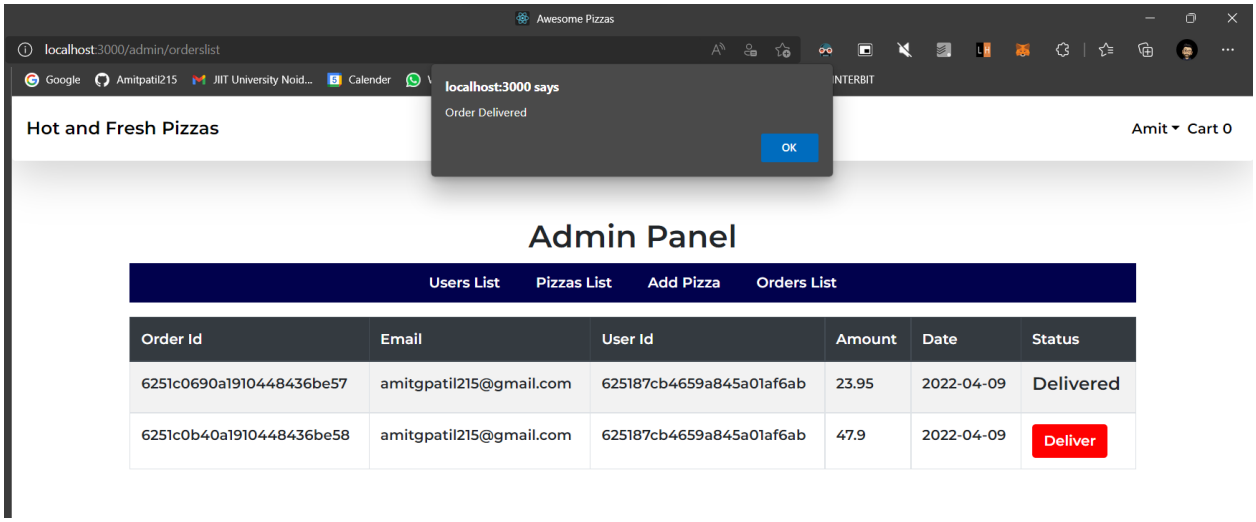



| Name | Prices | Category | Actions |
|---|---|---|---|
| PEPPER BARBECUE CHICKEN | Small : 100<br>Medium : 200<br>Large : 300 | veg | |
| Non Veg Supreme | Small : 200<br>Medium : 350<br>Large : 400 | nonveg | |
| Golden Corn Pizza | Small : 180<br>Medium : 250<br>Large : 400 | veg | |
| Jalapeno & Red Paprika Pizza | Small : 200<br>Medium : 300<br>Large : 400 | veg | |
| Margerita | Small : 150<br>Medium : 220<br>Large : 300 | veg | |
| Double Cheese Margherita Pizza | Small : 250<br>Medium : 350<br>Large : 399 | veg | |

Add Pizza Model



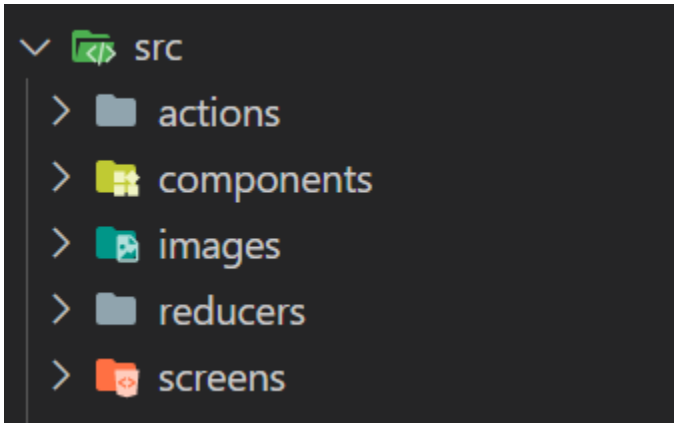

Order List




**Implementation details**

# Fronted

## 1. Project Structure




## 2. Home Screen

```
import React, { useState, useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { getAllPizzas } from "../actions/pizzaActions";
import Error from "../components/Error";
import Filter from "../components/Filter";
import Loading from "../components/Loading";
import Pizza from "../components/Pizza";
export default function Homescreen() {
  const dispatch = useDispatch();
```

```
  const pizzasstate = useSelector((state) => state.getAllPizzasReducer);

  const { pizzas, error, loading } = pizzasstate;

  useEffect(() => {
    dispatch(getAllPizzas());
  }, []);

  return (
    <div>
 <Filter/>
      <div className="row justify-content-center">

        {loading ? (
          <Loading/>
        ) : error ? (
          <Error error='Something went wrong'/>
        ) : (
          pizzas.map((pizza) => {
            return (
              <div className="col-md-3 m-3" key={pizza._id}>
                <div>
                  <Pizza pizza={pizza} />
                </div>
              </div>
            );
          })
        )}
      </div>
    </div>
  );
}
```

Login Screen

```
export default function Loginscreen() {
  const [email, setemail] = useState("");
  const [password, setpassword] = useState("");
  const loginstate = useSelector((state) => state.loginUserReducer);
```

```
  const { loading, error } = loginstate;
  const dispatch = useDispatch();

  useEffect(() => {
    if (localStorage.getItem("currentUser")) {
      window.location.href = "/";
    }
  }, []);

  function login() {
    const user = { email, password };
    dispatch(loginUser(user));
  }

  return (
    <div className="login">
      <div className="row justify-content-center mt-5">
        <div className="col-md-5 mt-5 text-left shadow-lg p-3 mb-5
bg-white rounded">
          <h2 className="text-center m-2" style={{ fontSize: "35px" }}>
            Login
          </h2>

          {loading && <Loading />}
          {error && <Error error="Invalid Credentials" />}

          <div>
            <input
              required
              type="email"
              placeholder="email"
              className="form-control"
              value={email}
              onChange={(e) => {
                setemail(e.target.value);
              }}
            />
            <input
              type="password"
              placeholder="password"
```

```
              className="form-control"
              value={password}
              required
              onChange={(e) => {
                setpassword(e.target.value);
              }}
            />

            <button onClick={login} className="btn mt-3 mb-3">
              LOGIN
            </button>
            <br />
            <a style={{ color: "black" }} href="/register"
className="mt-2">
              Click Here To Register
            </a>
          </div>
        </div>
      </div>
    </div>
  );
}
```

Pizza List

```
export default function Pizzaslist() {
  const dispatch = useDispatch();

  const pizzasstate = useSelector((state) => state.getAllPizzasReducer);

  const { pizzas, error, loading } = pizzasstate;
  useEffect(() => {
    dispatch(getAllPizzas());
  }, []);
  return <div>
    <h2>Pizzas List</h2>
    {loading && (<Loading/>)}
    {error && (<Error error='Something went wrong'/>)}
```

```
    <table  className='table table-bordered table-responsive-sm'>

        <thead className='thead-dark'>
            <tr>
                <th>Name</th>
                <th>Prices</th>
                <th>Category</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
        {pizzas && pizzas.map(pizza=>{

            return <tr>
                <td>{pizza.name}</td>
                <td>

                   Small : {pizza.prices[0]['small']} <br/>
                   Medium : {pizza.prices[0]['medium']} <br/>
                   Large : {pizza.prices[0]['large']}

                </td>
                <td>{pizza.category}</td>
                <td>
                    <i className='fa fa-trash m-1'
onClick={()=>{dispatch(deletePizza(pizza._id))}}></i>
                    <Link to={`/admin/editpizza/${pizza._id}`}><i
className='fa fa-edit m-1'></i></Link>
                </td>

            </tr>

        })}
        </tbody>

    </table>


  </div>;
}
```

Admin Screen

```
export default function Adminscreen() {
  const userstate = useSelector((state) => state.loginUserReducer);
  const { currentUser } = userstate;
  const dispatch = useDispatch();
  console.log(currentUser.isAdmin);
  useEffect(() => {
    console.log("start");
    if (!currentUser.isAdmin) {
      console.log(currentUser.isAdmin);
      window.location.href = "/";
    }
    console.log("frm use effect");
  }, []);

  return (
    <div>
      <div className="row justify-content-center p-3">
        <div className="col-md-10">
          <h2 style={{ fontSize: "35px" }}>Admin Panel</h2>

          <ul className="adminfunctions">
            <li>
              <Link to={'/admin/userslist'} style={{ color: 'white'
}}>Users List</Link>
            </li>
            <li>
              <Link to={'/admin/pizzaslist'} style={{ color: 'white'
}}>Pizzas List</Link>
            </li>
            <li>
              <Link to={'/admin/addpizza'} style={{ color: 'white' }}>Add
Pizza</Link>
            </li>
            <li>
              <Link to={'/admin/orderslist'} style={{ color: 'white'
}}>Orders List</Link>
```

```
            </li>


          </ul>


          <Switch>
            <Route path="/admin" component={Userslist} exact />
            <Route path="/admin/userslist" component={Userslist} exact />
            <Route path="/admin/orderslist" component={Orderslist} exact
/>
            <Route path="/admin/pizzaslist" component={Pizzaslist} exact
/>
            <Route path="/admin/addpizza" component={Addpizza} exact />
            <Route path="/admin/editpizza/:pizzaid" component={Editpizza}
exact />
          </Switch>
        </div>
      </div>
    </div>
  );
}
```

Navbar Component

```
export default function Navbar() {
  const cartstate = useSelector((state) => state.cartReducer);
  const userstate = useSelector((state) => state.loginUserReducer);
  const { currentUser } = userstate;
  const isAdmin = currentUser?.isAdmin;
  const dispatch = useDispatch();
  return (
    <div>
      <nav className="navbar navbar-expand-lg shadow-lg p-3 mb-5 bg-white
rounded">
        <a className="navbar-brand" href="/">
          Hot and Fresh Pizzas
        </a>
        <button
```

```
          className="navbar-toggler"
          type="button"
          data-toggle="collapse"
          data-target="#navbarNav"
          aria-controls="navbarNav"
          aria-expanded="false"
          aria-label="Toggle navigation"
        >
          <span className="navbar-toggler-icon">
            <i style={{ color: "black" }} className="fas fa-bars"></i>
          </span>
        </button>
        <div className="collapse navbar-collapse" id="navbarNav">
          <ul className="navbar-nav ml-auto">
            {currentUser ? (
              <div className="dropdown mt-2">
                <a
                  style={{ color: "black" }}
                  className="dropdown-toggle"
                  type="button"
                  id="dropdownMenuButton"
                  data-toggle="dropdown"
                  aria-haspopup="true"
                  aria-expanded="false"
                >
                  {currentUser.name}
                </a>
                <div
                  className="dropdown-menu"
                  aria-labelledby="dropdownMenuButton"
                >
                  {isAdmin ? (
                    <li className="nav-item">
                      <a className="dropdown-item" href="/admin">
                        Dashboard
                      </a>
                    </li>
                  ) : null}
                  <a className="dropdown-item" href="/orders">
                    Orders
```

```
                </a>
                <a
                  className="dropdown-item"
                  href="#"
                  onClick={() => {
                    dispatch(logoutUser());
                  }}
                >
                  <li>Logout</li>
                </a>
              </div>
            </div>
          ) : (
            <li className="nav-item">
              <a className="nav-link" href="/login">
                Login
              </a>
            </li>
          )}

          <li className="nav-item">
            <a className="nav-link" href="/cart">
              Cart {cartstate.cartItems.length}
            </a>
          </li>
        </ul>
      </div>
    </nav>
  </div>
);
}
```

## Backend

Server.js

```
const express = require("express");

const Pizza = require('./models/pizzaModel')

const app = express();
const db = require("./db.js")
app.use(express.json());
const path = require('path')
const pizzasRoute = require('./routes/pizzasRoute')
const userRoute = require('./routes/userRoute')
const ordersRoute = require('./routes/ordersRoute')





app.use('/api/pizzas/', pizzasRoute)
app.use('/api/users/', userRoute)
app.use('/api/orders/', ordersRoute)


if (process.env.NODE_ENV === 'production') {
    app.use('/', express.static('client/build'))

    app.get('*', (req, res) => {

        res.sendFile(path.resolve(__dirname, 'client/build/index.html'))

    })
}





const port = process.env.PORT || 8000;

app.listen(port, () => `Server running on port port 🔥`)
```

```
module.exports = app
```

User Routes

```
router.post("/register", async(req, res) => {

    const {name , email , password} = req.body

    const newUser = new User({name , email , password})

    try {
        newUser.save()
        res.send('User Registered successfully')
    } catch (error) {
         return res.status(400).json({ message: error });
    }

});


router.post("/login", async(req, res) => {

    const {email , password} = req.body

    try {

        const user = await User.find({email , password})

        if(user.length > 0)
        {
            const currentUser = {
                name : user[0].name ,
                email : user[0].email,
                isAdmin : user[0].isAdmin,
                _id : user[0]._id
            }
            res.send(currentUser);
        }
```

```
        else{
            return res.status(400).json({ message: 'User Login Failed' });
        }

    } catch (error) {
           return res.status(400).json({ message: 'Something went weong'
});
    }

});


router.get("/getallusers", async(req, res) => {

    try {
        const users = await User.find({})
        res.send(users)
    } catch (error) {
        return res.status(400).json({ message: error });
    }

});

router.post("/deleteuser", async(req, res) => {

    const userid = req.body.userid

    try {
        await User.findOneAndDelete({_id : userid})
        res.send('User Deleted Successfully')
    } catch (error) {
        return res.status(400).json({ message: error });
    }

});



module.exports = router
```

Pizza Routes

```
router.get("/getallpizzas", async(req, res) => {

    try {
        const pizzas = await Pizza.find({})
        res.send(pizzas)
    } catch (error) {
        return res.status(400).json({ message: error });
    }

});

router.post("/addpizza", async(req, res) => {

    const pizza = req.body.pizza

   try {
    const newpizza = new Pizza({
        name : pizza.name,
        image :pizza.image,
        varients : ['small','medium','large'],
        description : pizza.description,
        category : pizza.category,
        prices : [pizza.prices]
    })
    await newpizza.save()
    res.send('New Pizza Added Successfully')
   } catch (error) {
       return res.status(400).json({ message: error });
   }

});

router.post("/getpizzabyid", async(req, res) => {

 const pizzaid = req.body.pizzaid

 try {
     const pizza = await Pizza.findOne({_id : pizzaid})
     res.send(pizza)
```

```
 } catch (error) {
     return res.status(400).json({ message: error });
 }

});

router.post("/editpizza", async(req, res) => {

    const editedpizza = req.body.editedpizza

    try {
        const pizza = await Pizza.findOne({_id : editedpizza._id})

        pizza.name= editedpizza.name,
        pizza.description= editedpizza.description,
        pizza.image= editedpizza.image,
        pizza.category=editedpizza.category,
        pizza.prices = [editedpizza.prices]

        await pizza.save()

        res.send('Pizza Details Edited successfully')

    } catch (error) {
        return res.status(400).json({ message: error });
    }

});

router.post("/deletepizza", async(req, res) => {

    const pizzaid = req.body.pizzaid

  try {
    await Pizza.findOneAndDelete({_id : pizzaid})
    res.send('Pizza Deleted successfully')
  } catch (error) {
      return res.status(400).json({ message: error });
  }
```

```
});




module.exports = router;
```

Order Routes

```
const stripe = require("stripe")(
  process.env.SECRET_KEY
);
const Order = require("../models/orderModel");
router.post("/placeorder", async (req, res) => {
  const { token, subtotal, currentUser, cartItems } = req.body;

  try {
    // stripe.setPublishableKey(process.env.PUBLISH_KEY);
    console.log(process.env.PUBLISH_KEY)
    const customer = await stripe.customers.create({
      email: token.email,
      source: token.id,
    });

    const payment = await stripe.paymentIntents.create(
      {
        amount: subtotal * 100,
        currency: "inr",
        payment_method_types: ['card'],
        // customer: customer.id,
        // receipt_email: token.email,
      },
      {
        idempotencyKey: uuidv4(),
      }
    );

    if (payment) {
      const neworder = new Order({
        name: currentUser.name,
```

```
        email: currentUser.email,
        userid: currentUser._id,
        orderItems: cartItems,
        orderAmount: subtotal,
        shippingAddress: {
          street: token.card.address_line1,
          city: token.card.address_city,
          country: token.card.address_country,
          pincode: token.card.address_zip,
        },
        transactionId: payment.source != undefined ? payment.source.id :
uuidv4(),
      });

      neworder.save();

      res.send("Order placed successfully");
      console.log(" occured in try ");
    } else {
      console.log("Error occured in try else");
      res.send("Payment failed");
    }
  } catch (error) {
    console.log("Error occured in cath");
    console.log(error)

    return res.status(400).json({ message: "Something went wrong" + error
});
  }
});

router.post("/getuserorders", async (req, res) => {
  const { userid } = req.body;
  try {
    const orders = await Order.find({ userid: userid }).sort({ _id: -1 });
    res.send(orders);
  } catch (error) {
    return res.status(400).json({ message: "Something went wrong" });
  }
});
```

```
router.get("/getallorders", async (req, res) => {
  try {
    const orders = await Order.find({});
    res.send(orders);
  } catch (error) {
    return res.status(400).json({ message: error });
  }
});

router.post("/deliverorder", async (req, res) => {
  const orderid = req.body.orderid;
  try {
    const order = await Order.findOne({ _id: orderid });
    order.isDelivered = true;
    await order.save();
    res.send("Order Delivered Successfully");
  } catch (error) {
    return res.status(400).json({ message: error });
  }
});

module.exports = router;
```

# Testing details

### Mongoose connection test

```
describe("Check For mongoose connection", function () {
    it("Connection Established", function (done) {

        // tells mongoose to use ES6 implementation of promises
        mongoose.Promise = global.Promise;
        const MONGODB_URI = process.env.DB_URI;
        mongoose.connect(MONGODB_URI);

        mongoose.connection
            .once('open', () => console.log('Connected!'))
```

```
            .on('error', (error) => {
                console.warn('Error : ', error);
            });

        // console.log(mongoose.connection)
        emptyObj = {};

expect(mongoose.connection.collections.empty).to.equal(emptyObj.empty)
        done()
    });
});
```

### Mongoose add user test

```
describe("User", function () {
    it("Create New User", function (done) {

        // tells mongoose to use ES6 implementation of promises
        mongoose.Promise = global.Promise;
        const MONGODB_URI = process.env.DB_URI;
        mongoose.connect(MONGODB_URI);

        mongoose.connection
            .once('open', () => console.log('Connected!'))
            .on('error', (error) => {
                console.warn('Error : ', error);
            });
        const { name, email, password } = { name: "user1", email:
"user1@gmail.com", password: "password" }

        const newUser = new User({ name, email, password })

        try {
            newUser.save()
            expect("done").to.equal("done");
        } catch (error) {
            expect("done").to.equal("failed");
        }
        done()
```

```
    });
});
```

## Test Results

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL                                node

> mern-pizza@1.0.0 test
> mocha


  Check For mongoose connection
    ✓ Connection Established

  User
    ✓ Create New User


  2 passing (21ms)

PS E:\Work\Projects\awesomepizza> npm test
```

## References


1. (PDF) ONLINE FOOD ORDERING SYSTEM
2. Online Food Ordering System
3. Realtime pizza order tracker app using NodeJs, Express and Mongo DB in 2020 in Hindi. Introduction