SAVITRIBAI PHULE PUNE UNIVERSITY

A
MINI-PROJECT REPORT

ON

“ FOOD-ORDERING APPLICATION”

***(A project to fulfill the requirements of LP-II Lab*)**

By

ANUPAM KOSHTI	**307A009**
NANDINI BANGAD	**307A011**
SHUBHAM BIDGAR	**307A021**
NIDHI BHANDARI	**307A018**

Under the guidance of
(Prof. R. S. Vaidya)

DEPARTMENT OF INFORMATION TECHNOLOGY

SINHGAD COLLEEGE OF ENGINEERING, PUNE

(Academic Year: 2022 – 2023)

Sinhgad Institutes

DEPARTMENT OF INFORMATION TECHNOLOGY

SINHGAD COLLEGE OF ENGINEERING VADGAON, PUNE

<u>CERTIFICATE</u>

This is to certify that final project work entitled **"Food Ordering Application"** was successfully carried by

1. Anupam Koshti
2. Nandini Bangad
3. Shubham Bidgar
4. Nidhi Bhandari

In the partial fulfillment of the LABORATORY PRACTICE-II course during Semester-II of Third Year of Information Technology prescribed by the SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE.

Guide **H.O.D**

(Prof. R. S. Vaidya) **(Dr. S. R. Ganorkar)**

Principal

(Dr. S. D. Lokhande)

ACKNOWLEDGEMENT

I feel great pleasure in expressing my deepest sense of gratitude and sincere thanks to my guide Prof. R. S. Vaidya for their valuable guidance during the Project work, without which it would have been very difficult task. I have no words to express my sincere thanks for valuable guidance, extreme assistance and cooperation extended to all the Staff Members of department of Information Technology. This acknowledgement would be incomplete without expressing my special thanks to Dr. S. R.Ganorkar Head of the Department (Information Technology) for their support during the work.

I would also like to extend my heartfelt gratitude to my Principal, Dr. S. D. Lokhande who provided a lot of valuable support, mostly being behind the veils of college bureaucracy.

Last but not least I would like to thanks all the Teaching, Non- Teaching staff members of my Department, my parent and my colleagues those who helped me directly or indirectly for completing of this Project successfully.

Name of Students

ANUPAM KOSHTI	**307A009**
NANDINI BANGAD	**307A011**
SHUBHAM BIDGAR	**307A021**
NIDHI BHANDARI	**307A018**

ABSTRACT

With the increasing popularity of online food ordering, the demand for easy-to-use and user-friendly food order web applications has increased significantly. This report presents the development of a food order web application for a restaurant called New Relax Family Restaurant. The purpose of this project was to design and develop a web application that would enable customers to place food orders online, thereby increasing the efficiency of the ordering process and improving the customer experience.

The food order web application was developed using various web development technologies, including HTML, CSS, JavaScript, and ReactJS. The application allows customers to browse the menu, add items to their cart, and submit orders online. The application also includes a user registration and login system, allowing customers to create accounts, save their favorite items, and view their order history.

The development of the food order web application involved several phases, including requirement analysis, design, implementation, testing, and deployment. The project was completed within the given time frame, and the final application was tested for functionality and usability.

TABLE OF CONTENT

1.Project Problem statement: -

The traditional process of ordering food in restaurants can often be time-consuming and inconvenient for customers, leading to decreased customer satisfaction and potential loss of business for the restaurant. Moreover, with the increasing popularity of online food ordering, it has become necessary for restaurants to provide customers with an easy-to-use and user-friendly web application for placing food orders online.

To address this problem, this project aims to design and develop a food order web application for a restaurant called New Relax Family Restaurant. The objective is to develop an application that would enable customers to place food orders online, thereby increasing the efficiency of the ordering process and improving the customer experience. The web application should provide an easy-to-use and intuitive interface for customers, allowing them to browse the menu, add items to their cart, and submit orders online.

The development of the food order web application will involve various web development technologies, including HTML, CSS, JavaScript, and PHP. The application will also include a user registration and login system, allowing customers to create accounts, save their favorite items, and view their order history. The project will involve several phases, including requirement analysis, design, implementation, testing, and deployment

2. Software Requirements and Specification

- An updated Web Browser
- Internet Connectivity

2.1 Introduction

A food order web application built with React is an efficient and user-friendly platform that allows customers to order food online. With React, developers can create interactive and responsive user interfaces that provide seamless navigation and fast page load times.

The application can feature a search functionality that helps users find the food they crave quickly. It can also display restaurant menus, images, and pricing to provide a comprehensive ordering experience. Moreover, the application can enable customers to customize their orders with special instructions and dietary preferences.

With the help of React's advanced state management, developers can create a real-time order tracking feature, allowing customers to track their orders from preparation to delivery. Additionally, the application can feature a secure payment gateway that ensures customer transactions are safe and reliable.

Overall, a food order web application built with React is an excellent solution for restaurants and food businesses looking to expand their reach and streamline their operations. It is a convenient and user-friendly platform that can enhance the customer experience and boost sales.

2.2 Scope

A food ordering application has a broad scope, as it can serve different types of users and businesses in the food industry. Here are some examples of the scope of a food ordering application:

1. Customers: Food ordering apps provide a convenient way for customers to order their favorite meals from their favorite restaurants. Customers can easily browse menus, place orders, and pay for their meals through the app.

2. Restaurants: Restaurants can use food ordering apps to increase their sales by offering their menus to a wider audience. They can receive orders and payments through the app, manage their inventory, and track their sales and revenue.

3. Delivery services: Food ordering apps can also work as a platform for delivery services, connecting restaurants with customers who want their food delivered to their doorstep. Delivery services can use the app to manage their fleet of drivers, track deliveries, and handle payments.

4. Food bloggers and influencers: Food ordering apps can be used by food bloggers and influencers to recommend their favorite restaurants and dishes to their followers. They can use the app to create curated lists of restaurants and dishes, and earn commissions on sales made through the app.

5. Food events and festivals: Food ordering apps can be used to promote and sell tickets for food events and festivals, such as food fairs, tastings, and cooking classes.

In summary, a food ordering application has a wide scope and can serve different types of users and businesses in the food industry, including customers, restaurants, delivery services, food bloggers and influencers, and food events and festivals.

3. Approach

In this "food-ordering application" project we have used our own database. API will give the information like list of meals to be rendered on the screen which are static data and after submitting the order form the data is sent to same database.

4. Functionality of the code:

Front Page

The main page of this application is the front page. It consists of the available meals in the restaurant and a total items in the cart display. Users can add items from the menu and they can see the total amount and items in the cart.

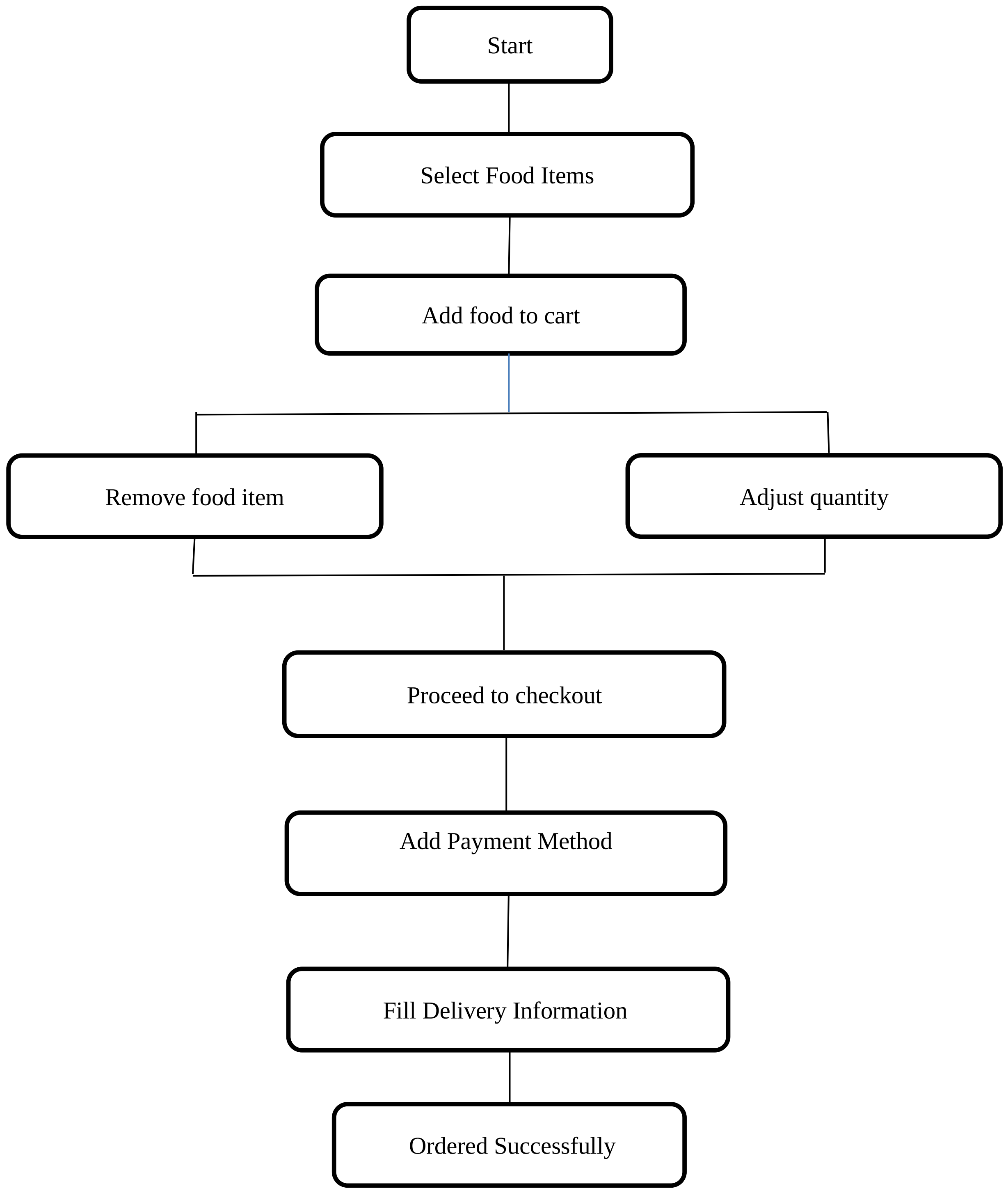
The Cart

In this section, users can see total amount and the quantity of the selected items before they can finally order them. They can add or remove the cart items accordingly from this section.

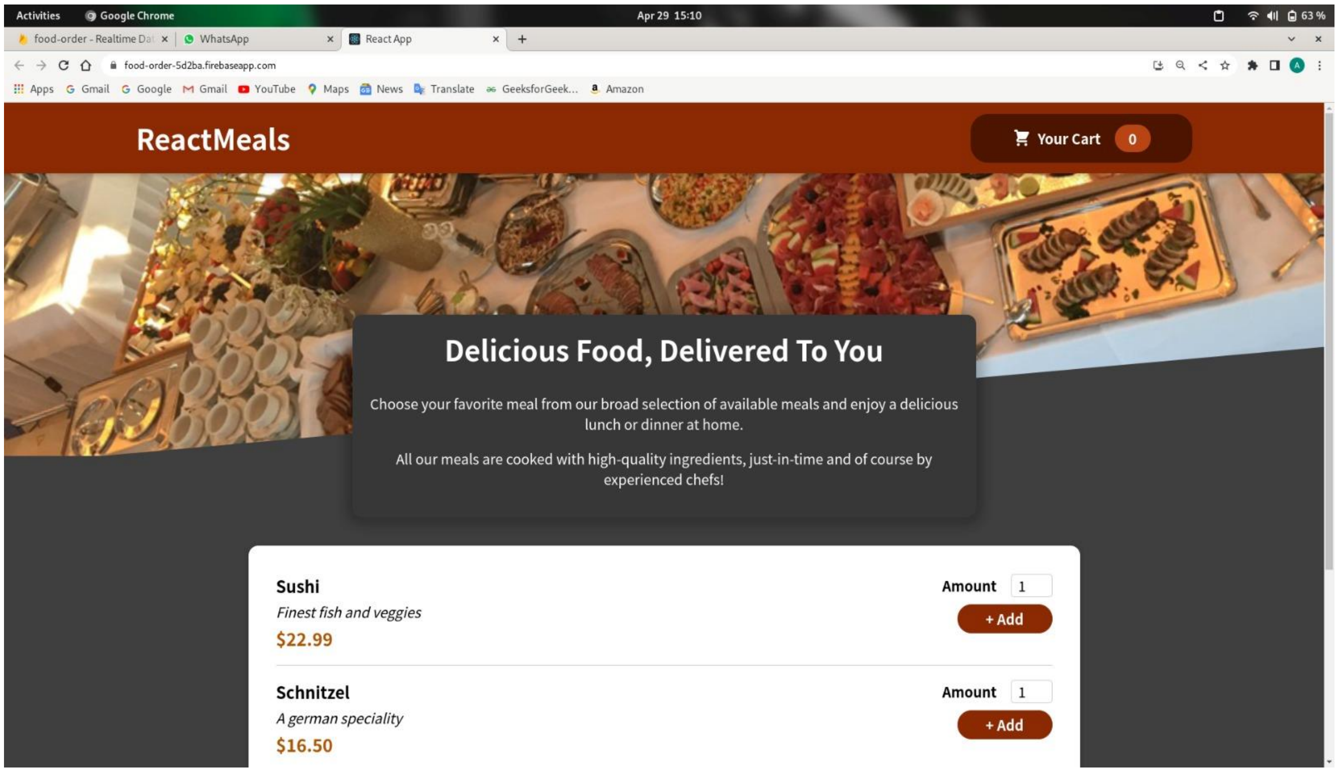
Orders

In this section, users have to give their address details with proper validation and make sure every input is properly filled according to the data types. After ordering they get a confirmation feedback message if the order was successfully placed or not.

5. FLOWCHART


Start
Select Food Items
Add food to cart
Remove food item
Adjust quantity
Proceed to checkout
Add Payment Method
Fill Delivery Information
Ordered Successfully

6. Project Outcomes:


Activities
Google Chrome
Apr 29 15:10
food-order - Realtime Dat
WhatsApp
React App
food-order-5d2ba.firebaseapp.com
Apps Gmail Google Gmail YouTube Maps News Translate GeeksforGeek... Amazon
Your Cart 0
Amount 1
+ Add

ReactMeals

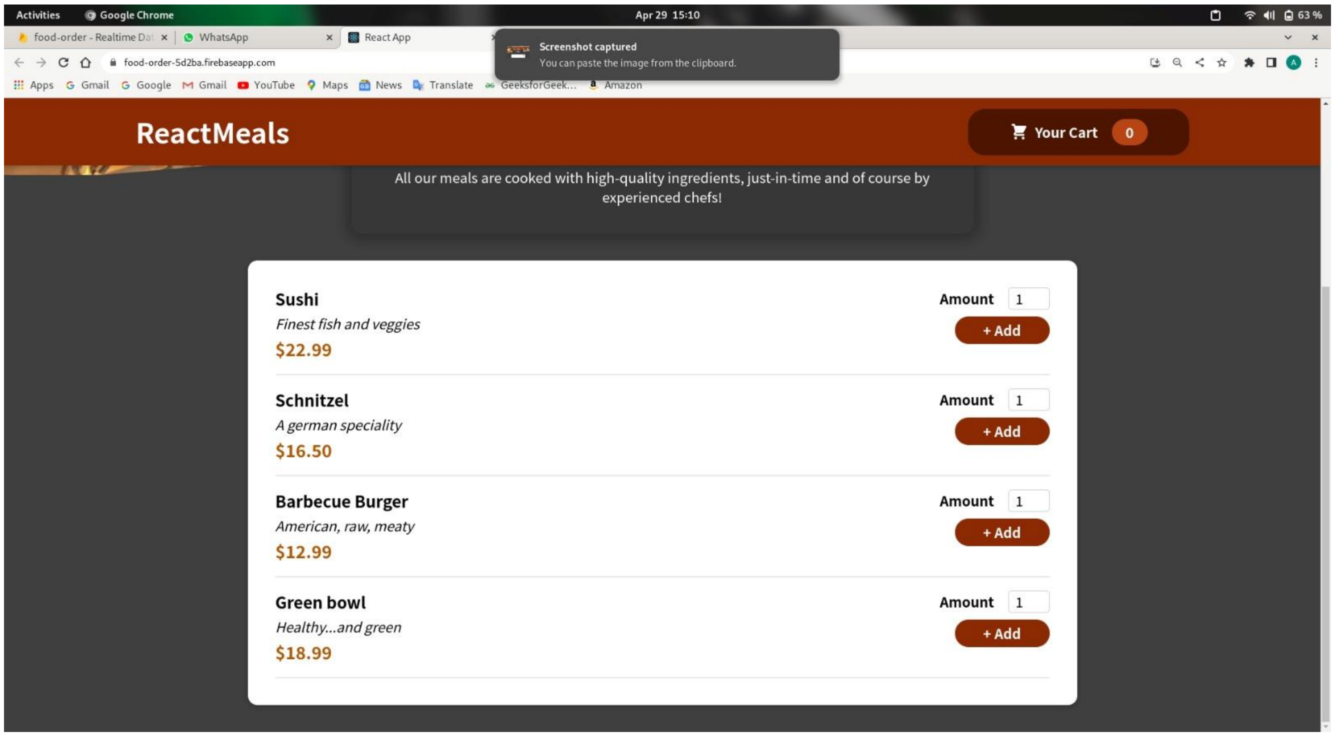
Delicious Food, Delivered To You

Choose your favorite meal from our broad selection of available meals and enjoy a delicious lunch or dinner at home.

All our meals are cooked with high-quality ingredients, just-in-time and of course by experienced chefs!

Sushi
Finest fish and veggies
$22.99

Schnitzel
A german speciality
$16.50


Activities
Google Chrome
Apr 29 15:10
food-order - Realtime Dat
WhatsApp
React App
Screenshot captured
You can paste the image from the clipboard.
food-order-5d2ba.firebaseapp.com
Apps Gmail Google Gmail YouTube Maps News Translate GeeksforGeek... Amazon
Your Cart 0
Amount 1
+ Add

ReactMeals

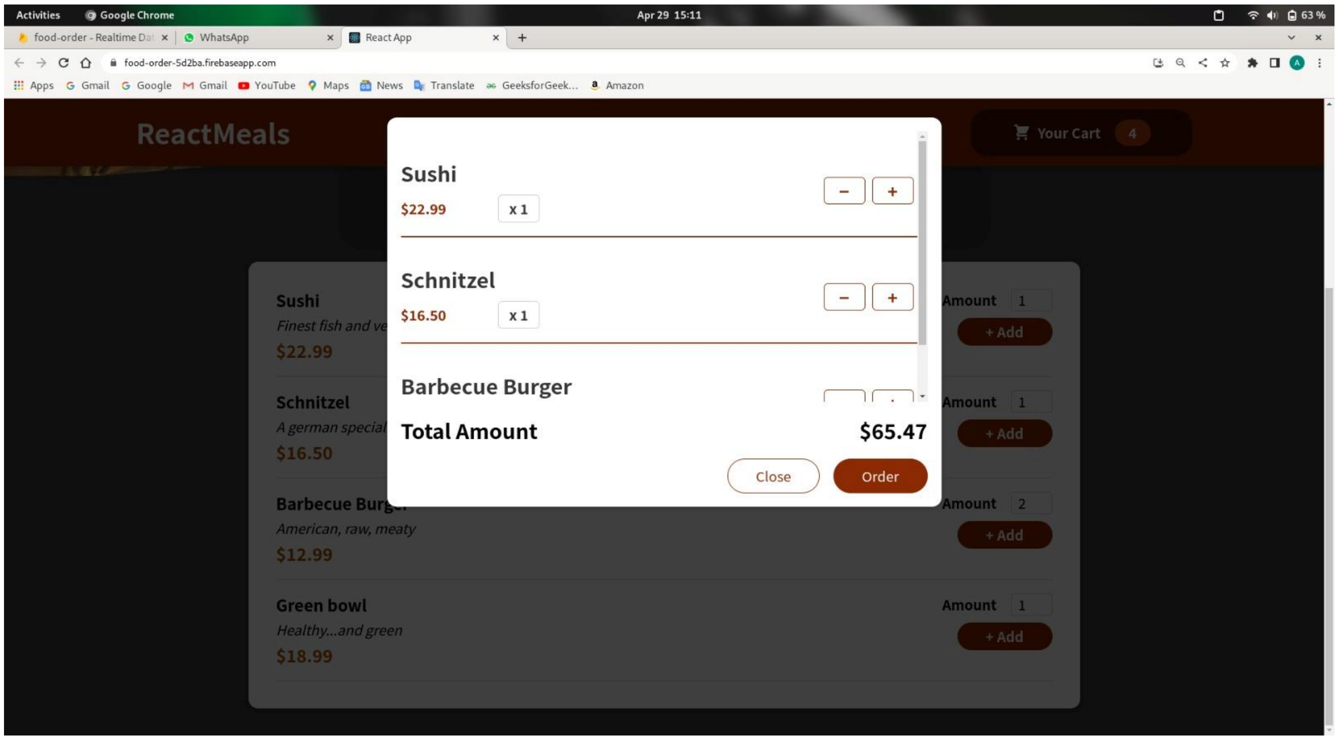
All our meals are cooked with high-quality ingredients, just-in-time and of course by experienced chefs!

Sushi
Finest fish and veggies
$22.99

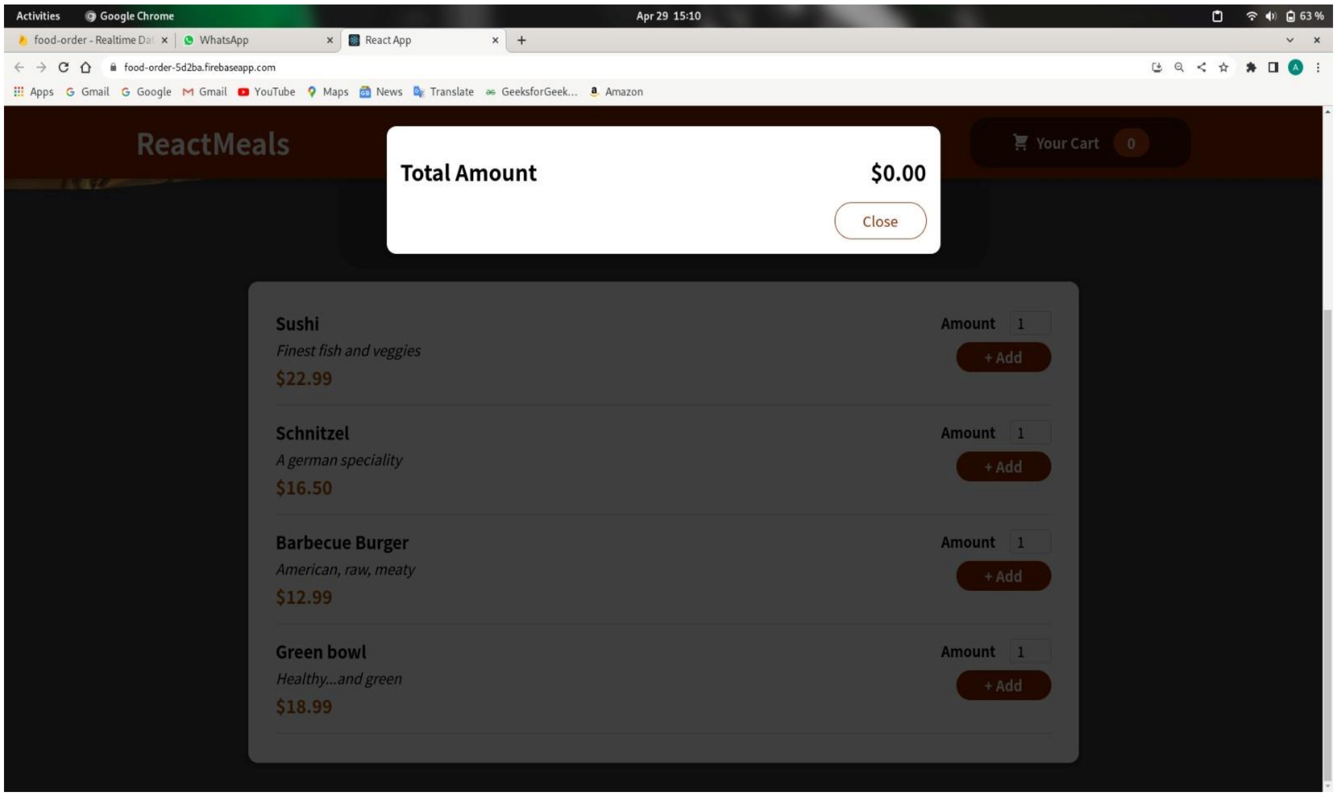
Schnitzel
A german speciality
$16.50

Barbecue Burger
American, raw, meaty
$12.99

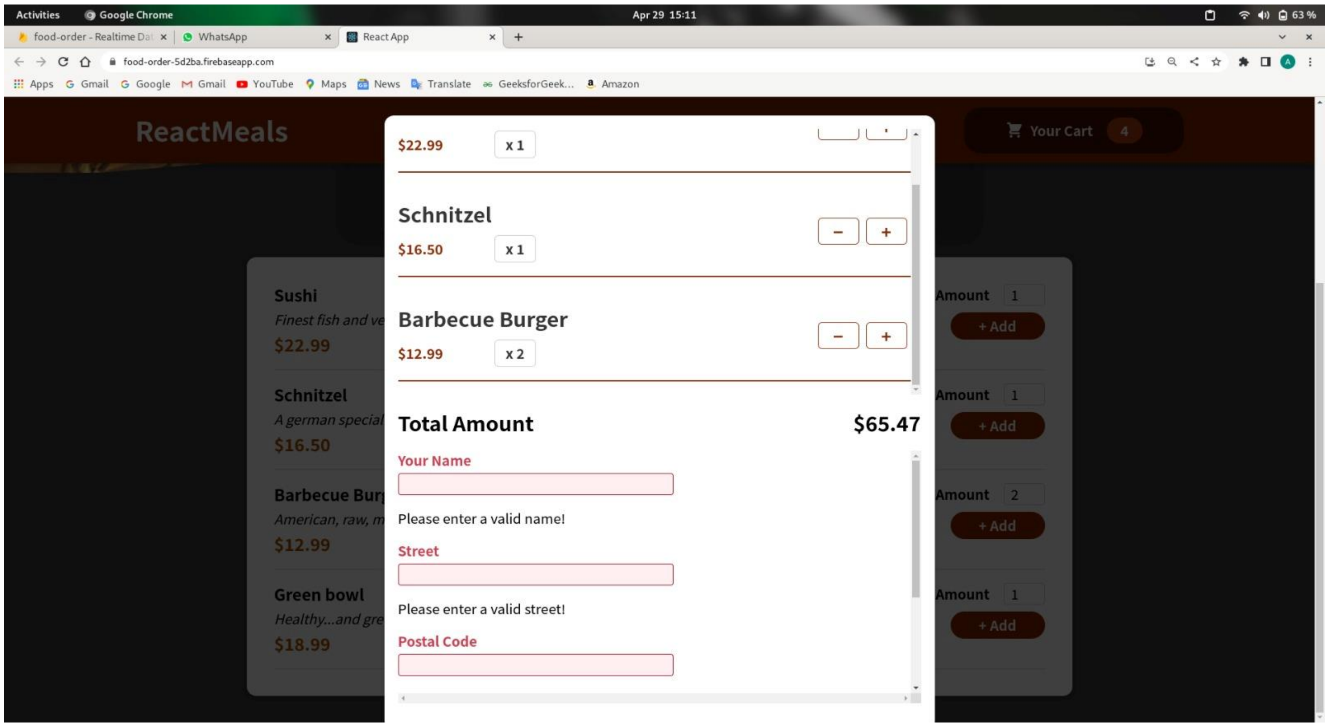
Green bowl
Healthy...and green
$18.99


Sushi
$22.99 x 1
Schnitzel
$16.50 x 1
Barbecue Burger
Total Amount $65.47

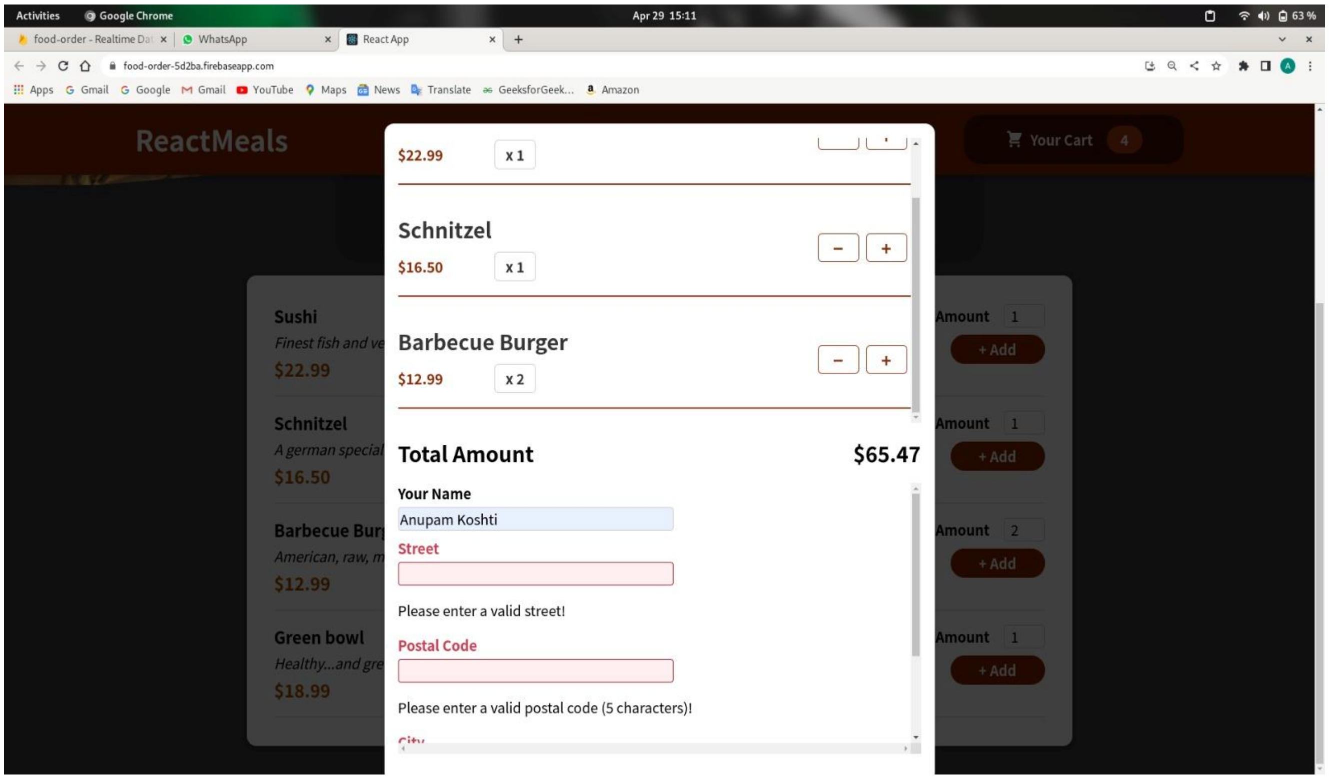
Sushi
Finest fish and veggies
$22.99
Schnitzel
A german speciality
$16.50
Barbecue Burger
American, raw, meaty
$12.99
Green bowl
Healthy...and green
$18.99


Total Amount $0.00

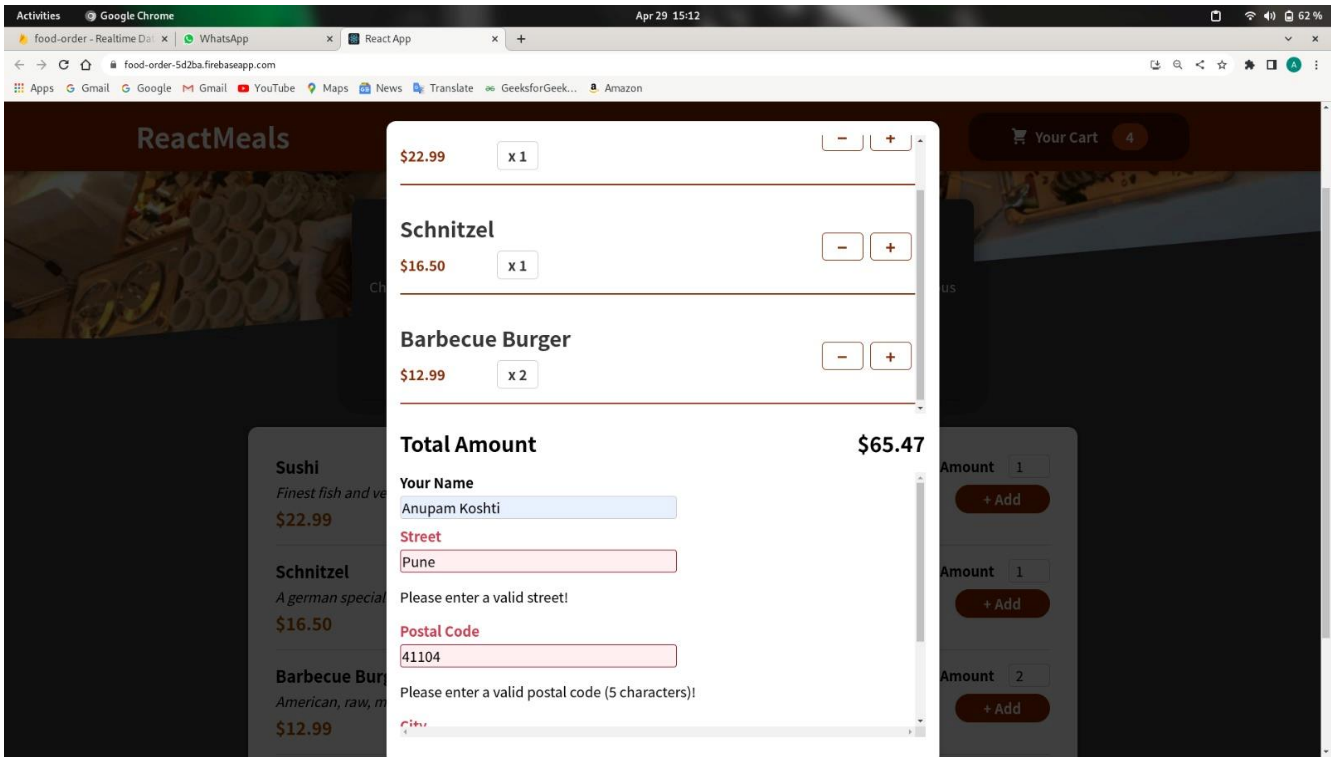
Sushi
Finest fish and veggies
$22.99
Schnitzel
A german speciality
$16.50
Barbecue Burger
American, raw, meaty
$12.99
Green bowl
Healthy...and green
$18.99



ReactMeals
Your Cart 4
$22.99 x 1
Schnitzel
$16.50 x 1
Barbecue Burger
$12.99 x 2
Total Amount $65.47
Your Name
Please enter a valid name!
Street
Please enter a valid street!
Postal Code



ReactMeals
Your Cart 4
$22.99 x 1
Schnitzel
$16.50 x 1
Barbecue Burger
$12.99 x 2
Total Amount $65.47
Your Name
Anupam Koshti
Street
Please enter a valid street!
Postal Code
Please enter a valid postal code (5 characters)!



$22.99 x 1

Schnitzel
$16.50 x 1

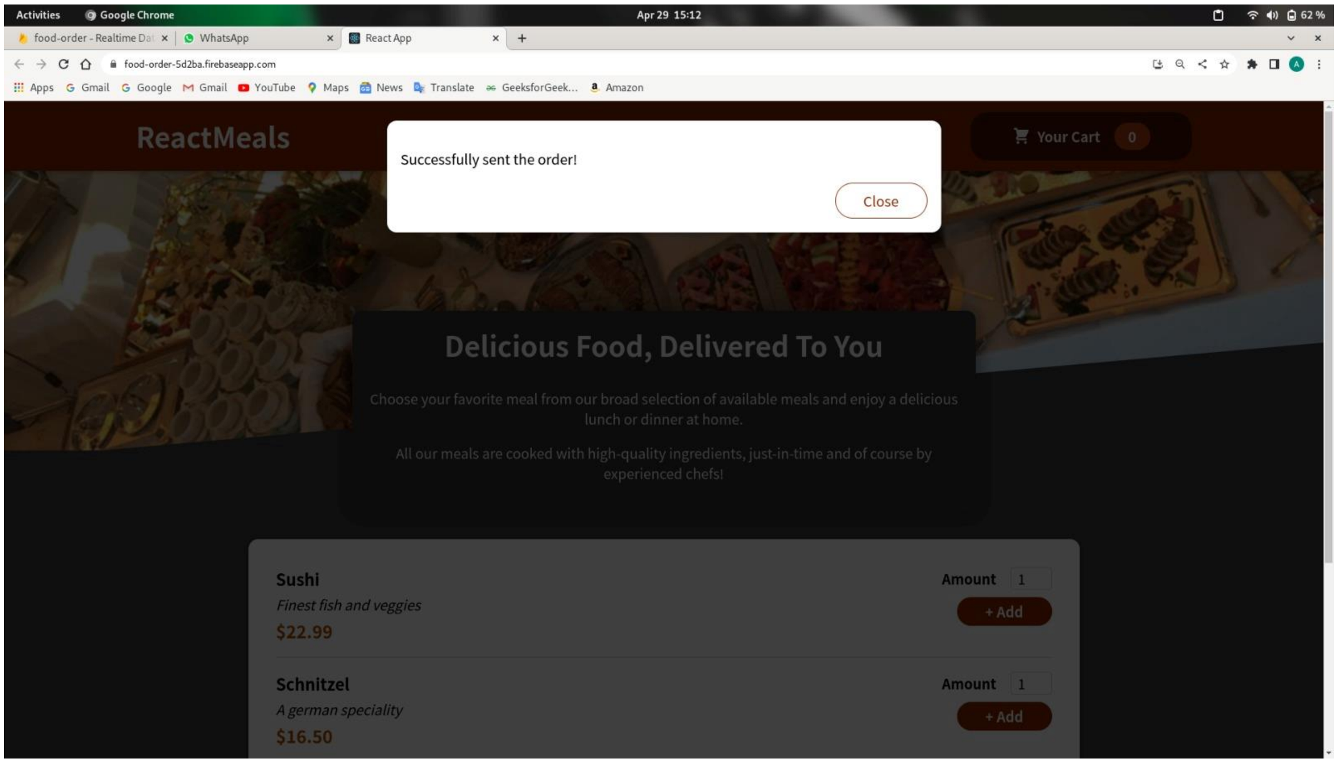
Barbecue Burger
$12.99 x 2

Total Amount $65.47

Your Name
Anupam Koshti

Street
Pune
Please enter a valid street!

Postal Code
41104
Please enter a valid postal code (5 characters)!



Successfully sent the order!

Close

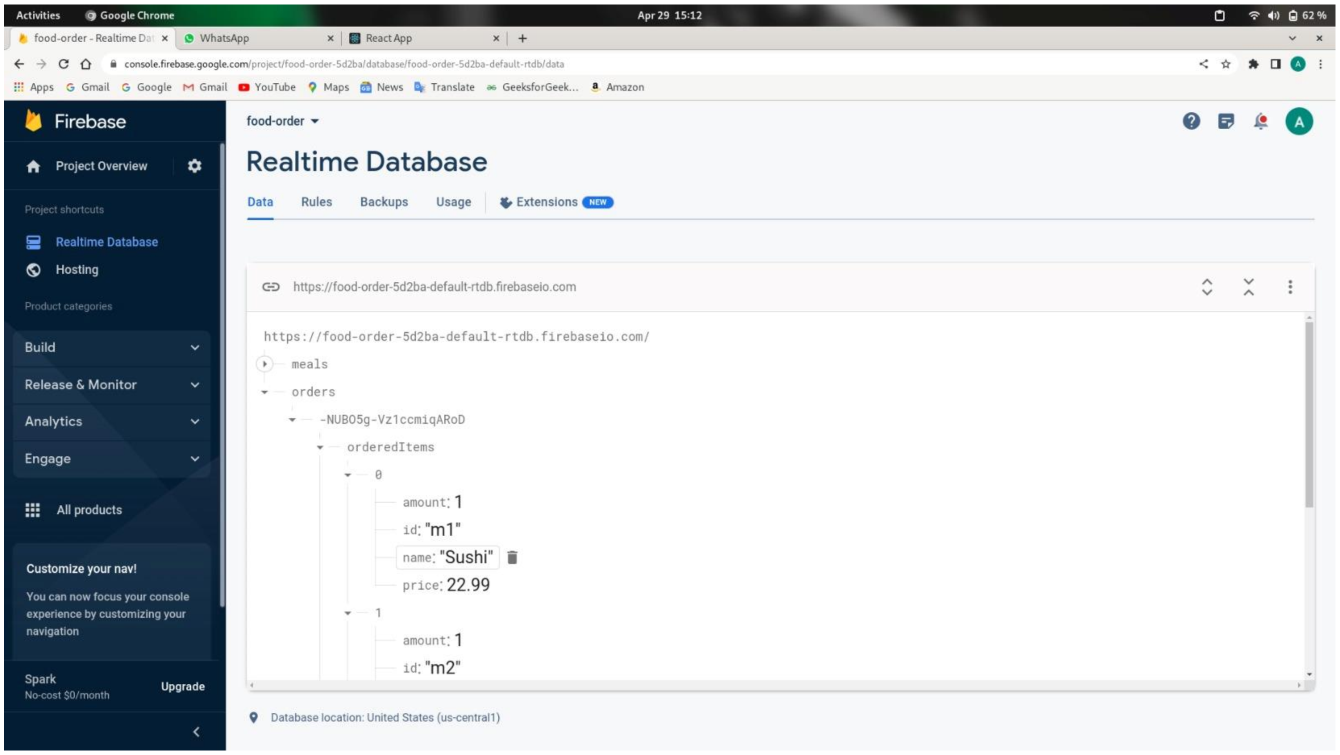
Delicious Food, Delivered To You

Choose your favorite meal from our broad selection of available meals and enjoy a delicious lunch or dinner at home.

All our meals are cooked with high-quality ingredients, just-in-time and of course by experienced chefs!

Sushi
Finest fish and veggies
$22.99
Amount 1
+ Add

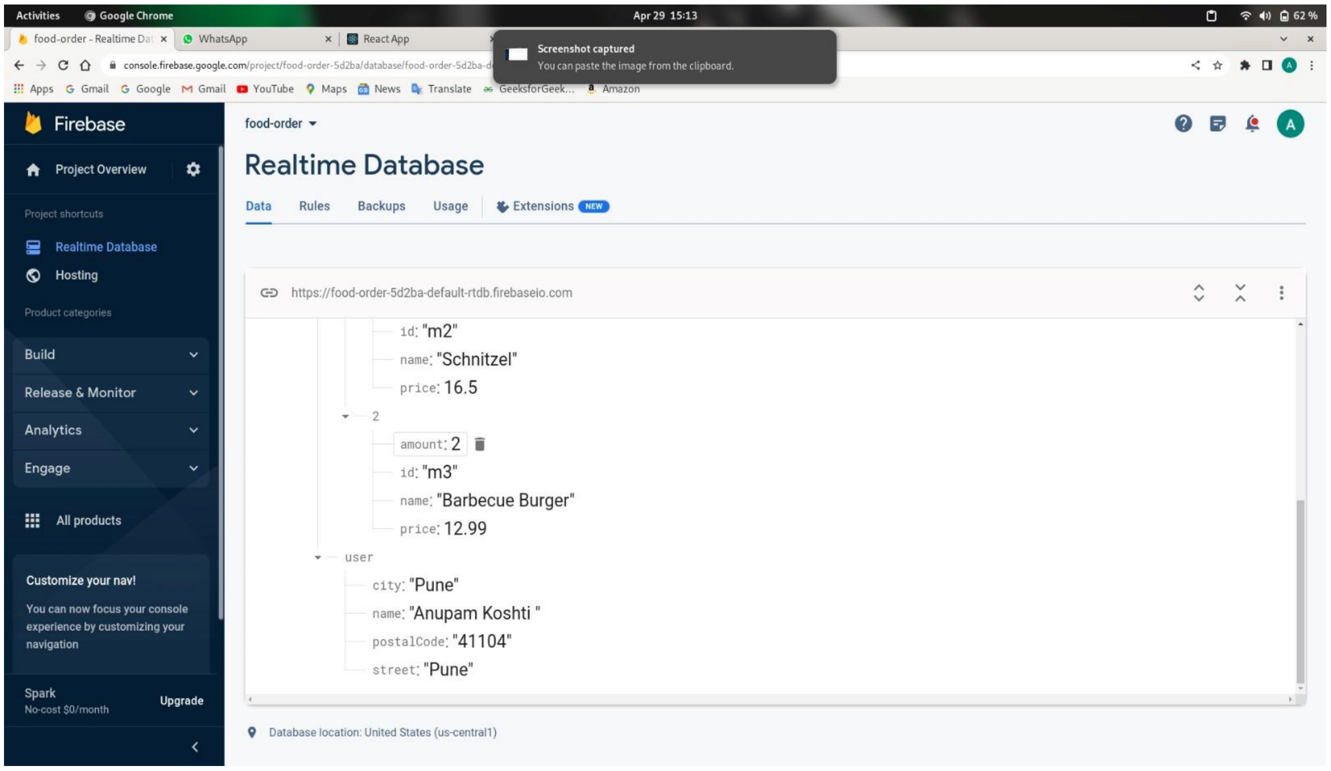
Schnitzel
A german speciality
$16.50
Amount 1
+ Add



https://food-order-5d2ba-default-rtdb.firebaseio.com

```
https://food-order-5d2ba-default-rtdb.firebaseio.com/
 meals
 orders
   -NUBO5g-Vz1ccmiqARoD
     orderedItems
       0
         amount: 1
         id: "m1"
         name: "Sushi"
         price: 22.99
       1
         amount: 1
         id: "m2"
```

Database location: United States (us-central1)



https://food-order-5d2ba-default-rtdb.firebaseio.com

```
         id: "m2"
         name: "Schnitzel"
         price: 16.5
       2
         amount: 2
         id: "m3"
         name: "Barbecue Burger"
         price: 12.99
     user
       city: "Pune"
       name: "Anupam Koshti "
       postalCode: "41104"
       street: "Pune"
```

Database location: United States (us-central1)

7.Source Code:

App.js

```
import { useState } from "react";
import Cart from "./components/Cart/Cart";
import Header from "./components/Layout/Header";
import Meals from "./components/Meals/Meals";
import CartProvider from "./store/CartProvider";

function App() {
  const [cartIsShown, setCartIsShown] = useState(false);

  const showCartHandler = () => {
    setCartIsShown(true);
  };
  const hideCartHandler = () => {
    setCartIsShown(false);
  };

  return (
    <CartProvider>
      {/* if cartIsShown is true then only show cart */}
      {cartIsShown && <Cart onHideCart={hideCartHandler} />}
      <Header onShowCart={showCartHandler} />
      <main>
        <Meals />
      </main>
    </CartProvider>
  );
}

export default App;
```

AvailabelMeals.js

```
import { useEffect, useState } from "react";
import Card from "../UI/Card";
import classes from "./AvailableMeals.module.css";
import MealItem from "./MealItem/MealItem";

const AvailableMeals = () => {
  const [meals, setMeals] = useState([]);
  const [isLoading, setIsLoading] = useState(true);
const [httpError, setHttpError] = useState();

useEffect(() => {
  const fetchMeals = async () => {
    const response = await fetch(
      "https://food-order-5d2ba-default-rtdb.firebaseio.com/meals.json"
    );
    const responseData = await response.json();

    if (!response.ok) {
      throw new Error("Something went wrong!");
    }

    const loadedMeals = [];
    for (const key in responseData) {
      loadedMeals.push({
        id: key,
        name: responseData[key].name,
        description: responseData[key].description,
        price: responseData[key].price,
      });
    }

    setMeals(loadedMeals);
    // console.log(responseData);
    setIsLoading(false);
  };

  fetchMeals().catch((error) => {
    setIsLoading(false);
    setHttpError(error.message);
  });
}, []);

if (isLoading) {
  return (
    <section className={classes.MealsLoading}>
      <p>Loading...</p>
    </section>
  );
}

if (httpError) {
  return (
    <section className={classes.MealsError}>
      <p>{httpError}</p>
      </section>
    );
  }

  const mealsList = meals.map((meal) => (
    <MealItem
      key={meal.id}
      id={meal.id}
      price={meal.price}
      description={meal.description}
      name={meal.name}
    />
  ));

  return (
    <section className={classes.meals}>
      <Card>
        <ul>{mealsList}</ul>
      </Card>
    </section>
  );
};
export default AvailableMeals;
```

Cart.js

```
import { Fragment, useContext, useState } from "react";
import CartContext from "../../store/cart-context";
import Modal from "../UI/Modal";
import classes from "./Cart.module.css";
import CartItem from "./CartItem";
import Checkout from "./Checkout";

const Cart = (props) => {
  const [isCheckOut, setIsCheckOut] = useState(false);
  const [isSubmitting, setIsSubmitting] = useState(false);
  const [didSubmit, setDidSubmit] = useState(false);

  const cartCtx = useContext(CartContext);
  const totalAmount = `$${cartCtx.totalAmount.toFixed(2)}`;
  const hasItems = cartCtx.items.length > 0;
  //   console.log(cartCtx.totalAmount);
  //   console.log(typeof totalAmount);

  const cartItemAddHandler = (item) => {
```

```
  // cartCtx.addItem(item)

  //instead use
  cartCtx.addItem({
    ...item,
    amount: 1,
  });
};

const cartItemRemoveHandler = (id) => {
  cartCtx.removeItem(id);
};

const orderHandler = () => {
  setIsCheckOut(true);
};

const submitOrderHandler = async (userData) => {
  setIsSubmitting(true);
  const response = await fetch(
    "https://food-order-5d2ba-default-rtdb.firebaseio.com/orders.json",
    {
      method: "POST",
      body: JSON.stringify({
        user: userData,
        orderedItems: cartCtx.items,
      }),
    }
  );
  setIsSubmitting(false);
  setDidSubmit(true);
  cartCtx.clearCart();
};

const cartItems = (
  <ul className={classes["cart-items"]}>
    {cartCtx.items.map((item) => (
      <CartItem
        key={item.id}
        name={item.name}
        amount={item.amount}
        price={item.price}
        onRemove={cartItemRemoveHandler.bind(null, item.id)}
        onAdd={cartItemAddHandler.bind(null, item)}
      />
    ))}
  </ul>
);

const cartModalContent = (
  <Fragment>
    {cartItems}
    <div className={classes.total}>
      <span>Total Amount</span>
      <span>{totalAmount}</span>
    </div>

    {isCheckOut && (
      <Checkout onConfirm={submitOrderHandler} onCancel={props.onHideCart} />
    )}

    {!isCheckOut && (
      <div className={classes.actions}>
        <button onClick={props.onHideCart} className={classes["button--alt"]}>
          Close
        </button>
        {hasItems && (
          <button className={classes.button} onClick={orderHandler}>
            Order
          </button>
        )}
      </div>
    )}
  </Fragment>
);

const isSubmittingModalContent = <p>Sending order data... </p>;
const didSubmitModalContent = (
  <Fragment>
    <p>Successfully sent the order!</p>
    <div className={classes.actions}>
      <button onClick={props.onHideCart} className={classes["button--alt"]}>
        Close
      </button>
    </div>
  </Fragment>
);

return (
  <Modal onHideCart={props.onHideCart}>
    {!isSubmitting && !didSubmit && cartModalContent}
    {isSubmitting && isSubmittingModalContent}
```

```
      {!isSubmitting && didSubmit && didSubmitModalContent}
    </Modal>
  );
};

export default Cart;
```

Checkout.js

```
import { useRef, useState } from "react";
import classes from "./Checkout.module.css";

const isEmpty = (value) => value.trim() === "";
const isFiveChar = (value) => value.trim().length === 5;

const Checkout = (props) => {
  const [formInputsValidity, setFormInputsValidity] = useState({
    name: true,
    street: true,
    city: true,
    postalCode: true,
  });

  const nameInputRef = useRef();
  const streetInputRef = useRef();
  const postalInputRef = useRef();
  const cityInputRef = useRef();

  const confirmHandler = (event) => {
    event.preventDefault();

    const enteredName = nameInputRef.current.value;
    const enteredStreet = streetInputRef.current.value;
    const enteredPostalCode = postalInputRef.current.value;
    const enteredCity = cityInputRef.current.value;

    const enteredNameIsValid = !isEmpty(enteredName);
    const enteredStreetIsValid = !isEmpty(enteredStreet);
    const enteredCityIsValid = !isEmpty(enteredCity);
    const enteredPostalcodeIsValid = isFiveChar(enteredPostalCode);

    setFormInputsValidity({
      name: enteredNameIsValid,
      street: enteredStreetIsValid,
      city: enteredCityIsValid,
      postalCode: enteredPostalcodeIsValid,
    })

    const formIsValid =
      enteredNameIsValid &&
      enteredStreetIsValid &&
      enteredCityIsValid &&   enteredPostalcodeIsValid;
if (!formIsValid) {
      return;
    }

    //submit the form data to backend

    props.onConfirm({
        name: enteredName,
        street: enteredStreet,
        postalCode: enteredPostalCode,
        city: enteredCity
    })
  };

  const nameControlClasses = `${classes.control} ${formInputsValidity.name ? '' : classes.invalid}`;
  const streetControlClasses = `${classes.control} ${formInputsValidity.street ? '' : classes.invalid}`;
  const postalCodeControlClasses = `${classes.control} ${formInputsValidity.postalCode ? '' : classes.invalid}
  const cityControlClasses = `${classes.control} ${formInputsValidity.city ? '' : classes.invalid}`;

  return (
    <form className={classes.form} onSubmit={confirmHandler}>
      <div className={nameControlClasses}>
        <label htmlFor='name'>Your Name</label>
        <input type='text' id='name' ref={nameInputRef} />
        {!formInputsValidity.name && <p>Please enter a valid name!</p>}
      </div>
      <div className={streetControlClasses}>
        <label htmlFor='street'>Street</label>
        <input type='text' id='street' ref={streetInputRef} />
        {!formInputsValidity.street && <p>Please enter a valid street!</p>}

      </div>
      <div className={postalCodeControlClasses}>
        <label htmlFor='postal'>Postal Code</label>
        <input type='text' id='postal' ref={postalInputRef} />
        {!formInputsValidity.postalCode && <p>Please enter a valid postal code (5 characters)!</p>}
      </div>
      <div className={cityControlClasses}>
        <label htmlFor='city'>City</label>
        <input type='text' id='city' ref={cityInputRef} />
        {!formInputsValidity.city && <p>Please enter a valid city!</p>}
      </div>
      <div className={classes.actions}>
        <button type='button' onClick={props.onCancel}>
          Cancel
        </button>
        <button className={classes.submit}>Confirm</button>
      </div>
    </form>
  );

}

export default Checkout;
```

8.CONCLUSION

In conclusion, a food ordering application can be a convenient and efficient way for customers to order food from their favorite restaurants. With the help of technology, customers can browse menus, customize their orders, and track their delivery status in real-time. Restaurants can also benefit from such applications as they can increase their reach, streamline their operations, and offer a better customer experience. However, it is important to ensure the security of personal and financial information and to provide reliable and prompt delivery services to ensure customer satisfaction. Overall, a well-designed and executed food ordering application can be a win-win for both customers and restaurants.