FULL STACk PROJECT

(2022-23)

"Food Delivery WebApp"

Project Report

Institute of Engineering & Technology

Submitted By-
Gaurav Kumar Tiwari (201500250)
Isha Agrawal (201500304)
Madhav Maheshwari (201500378)

Under the supervision of

Mr. Akash Kumar Choudhary

Technical Trainer

Department of Computer Engineering & Applications

Department of Computer Engineering and Applications
GLA University, 17 km. Stone NH#2, Mathura-Delhi Road,
Chaumuha, Mathura – 281406 U.P (India)

Declaration

I/We hereby declare that the work which is being presented in the Bachelor of Technology. Project **"Food Delivery WebApp"**, in partial fulfilment of the requirements for the award of the ***Bachelor of Technology*** in Computer Science an Engineering and submitted to the Department of computer Engineering and Applications of GLA University, Mathura, is an authentic record of my/our own work carried under the supervision of **Mr. Akash Kumar Choudhary, Technical Trainer, GLA University.**

The contents of this project report, in full or in parts, have not been submitted to any other institute or University for the award of any degree.

Sign: Gaurav Kumar Tiwari

Name of Candidate: Gaurav Kumar Tiwari

University Roll No:201500250

Sign: Isha Agrawal

Name of Candidate: Isha Agrawal

University Roll No: 201500304

Sign: Madhav Maheshwari

Name of Candidate: Madhav Maheshwari

University Roll No: 201500378

Department of Computer Engineering and Applications
GLA University, 17 km. Stone NH#2, Mathura-Delhi Road,
Chaumuha, Mathura – 281406 U.P (India)

Certificate

This is to certify that the project entitled "Food Delivery webApp", carried out in Full Stack Project , is a Bonafede work by Gaurav Kumar Tiwari Isha Agrawal, Madhav Maheshwari and is submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Computer Science & Engineering).

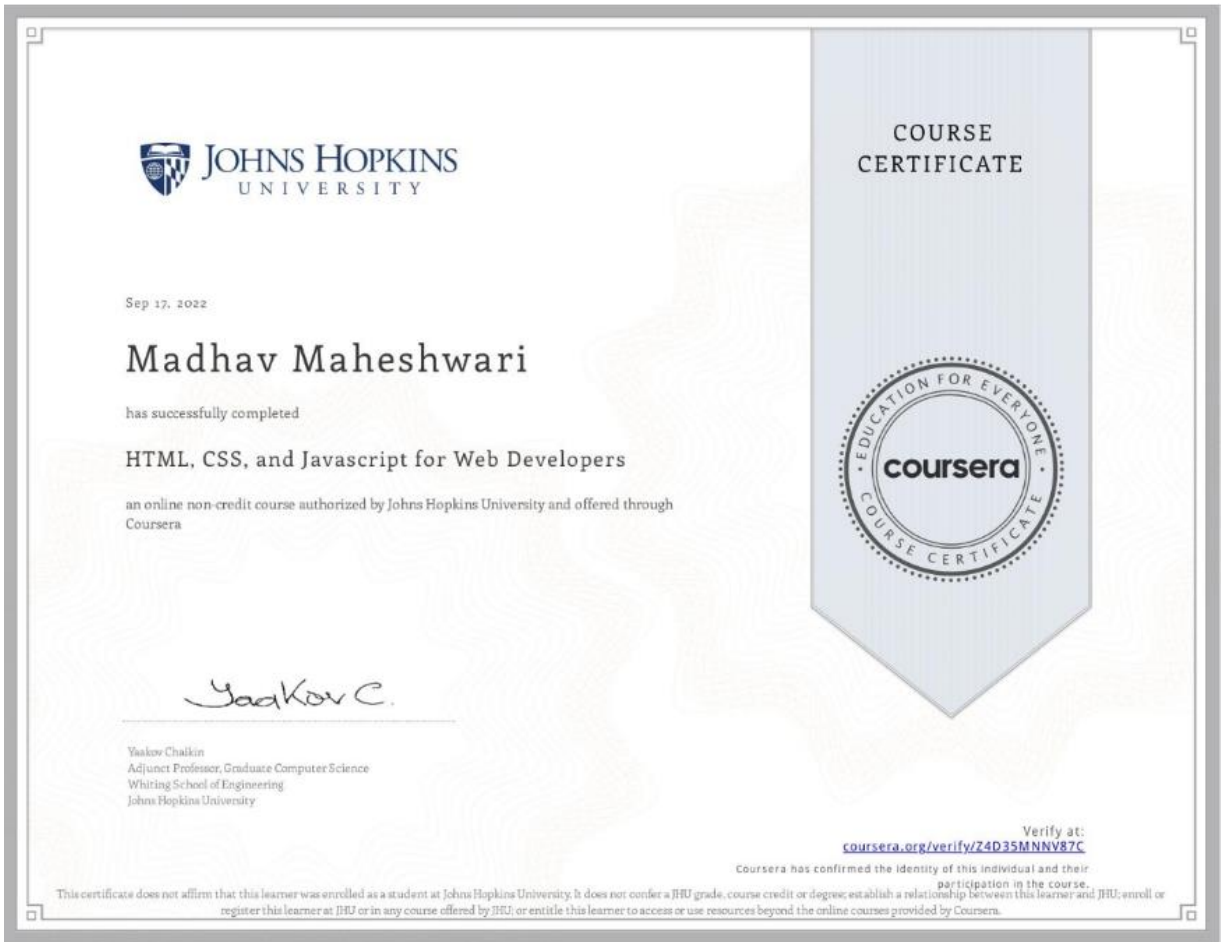
Signature of Supervisor:

Name of the Supervisor: Akash Kumar Choudhary

Date: April 26, 2023

Training Certificates

Madhav Mahesgwari



JOHNS HOPKINS
UNIVERSITY

COURSE
CERTIFICATE

Sep 17, 2022

Madhav Maheshwari

has successfully completed

HTML, CSS, and Javascript for Web Developers

an online non-credit course authorized by Johns Hopkins University and offered through Coursera

EDUCATION FOR EVERYONE
coursera
COURSE CERTIFICATE

YaaKov C.

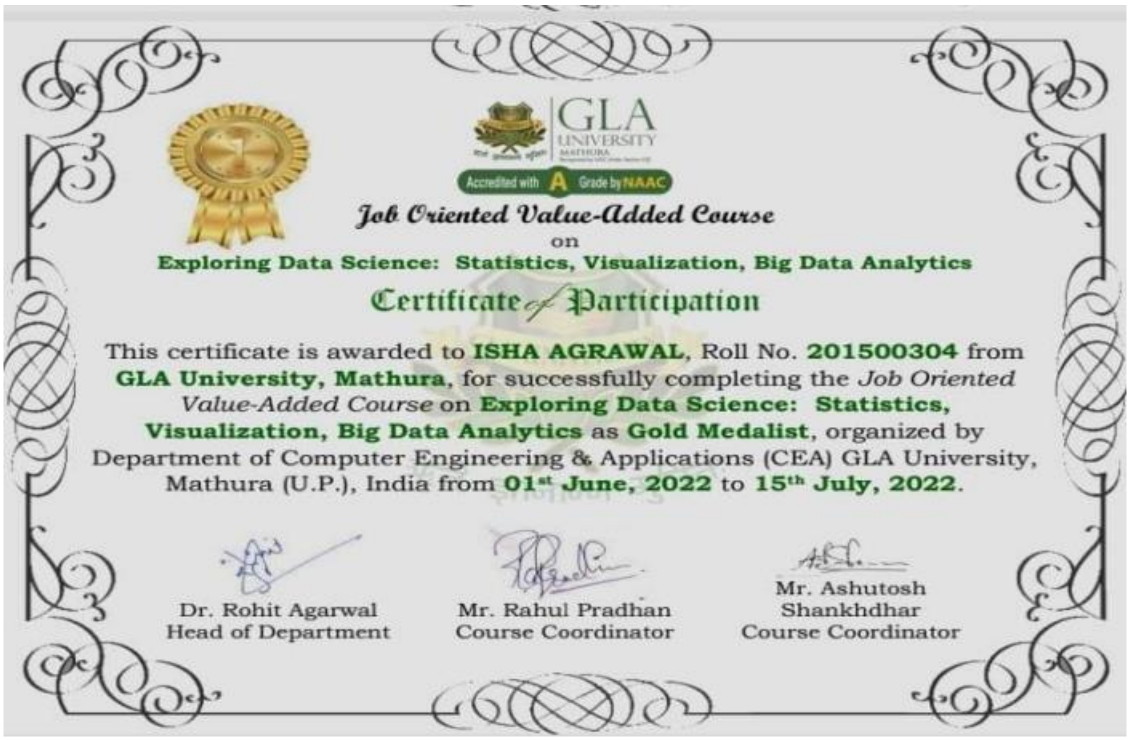
Yaakov Chaikin
Adjunct Professor, Graduate Computer Science
Whiting School of Engineering
Johns Hopkins University

Verify at:
coursera.org/verify/Z4D35MNNV87C

Coursera has confirmed the identity of this individual and their participation in the course.

This certificate does not affirm that this learner was enrolled as a student at Johns Hopkins University. It does not confer a JHU grade, course credit or degree; establish a relationship between this learner and JHU; enroll or register this learner at JHU or in any course offered by JHU; or entitle this learner to access or use resources beyond the online courses provided by Coursera.

Isha Agrawal



GLA
UNIVERSITY
MATHURA

Accredited with A Grade by NAAC

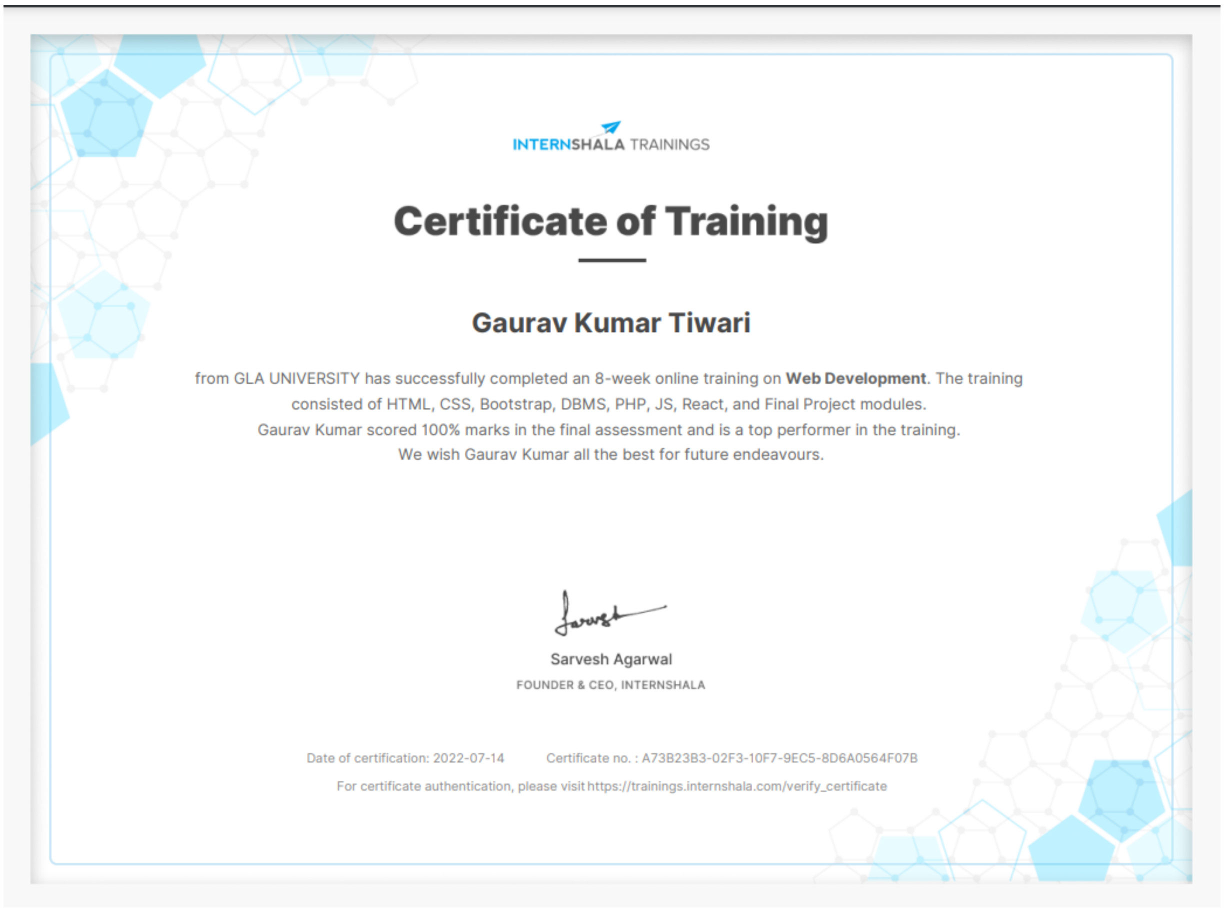
Job Oriented Value-Added Course
on
Exploring Data Science: Statistics, Visualization, Big Data Analytics

Certificate of Participation

This certificate is awarded to **ISHA AGRAWAL**, Roll No. **201500304** from **GLA University, Mathura**, for successfully completing the *Job Oriented Value-Added Course* on **Exploring Data Science: Statistics, Visualization, Big Data Analytics** as **Gold Medalist**, organized by Department of Computer Engineering & Applications (CEA) GLA University, Mathura (U.P.), India from **01st June, 2022** to **15th July, 2022**.

[illegible]	[illegible]	[illegible]
Dr. Rohit Agarwal	Mr. Rahul Pradhan	Mr. Ashutosh Shankhdhar
Head of Department	Course Coordinator	Course Coordinator

Gaurav Kumar Tiwari



INTERNSHALA TRAININGS

Certificate of Training

Gaurav Kumar Tiwari

from GLA UNIVERSITY has successfully completed an 8-week online training on **Web Development**. The training consisted of HTML, CSS, Bootstrap, DBMS, PHP, JS, React, and Final Project modules.
Gaurav Kumar scored 100% marks in the final assessment and is a top performer in the training.
We wish Gaurav Kumar all the best for future endeavours.

[illegible]

Sarvesh Agarwal
FOUNDER & CEO, INTERNSHALA

Date of certification: 2022-07-14 Certificate no. : A73B23B3-02F3-10F7-9EC5-8D6A0564F07B

For certificate authentication, please visit https://trainings.internshala.com/verify_certificate

ऋते ज्ञानान्न मुक्तिः

Department of Computer Engineering and Applications

GLA University, 17 km. Stone NH#2, Mathura-Delhi Road,

Chaumuha, Mathura – 281406 U.P (India)

ACKNOWLEDGEMENT

We would like to express my gratitude towards Mr. Akash Kumar Choudhary for guiding us throughout the project. We also feel thankful and express my kind gratitude towards all our teachers for allowing me to conduct Food Delivery WebApp project. The mentioned project was done under the supervision of Mr. Akash Kumar Choudhary.

He has been helping us since Day 1 in this project. He provided us with the roadmap, the basic guidelines explaining on how to work on the project. He has been conducting regular meeting to check the progress of the project and providing us with the resources related to the project. Without his help, we wouldn't have been able to complete this project.

We feel thankful to the college staff for giving me such a big opportunity. I believe We will enroll in more such events in the coming future.

Thanking You

Sign: Gaurav Kumar Tiwari

Name of Candidate: Gaurav Kumar Tiwari

University Roll No:201500250

Sign: Isha Agrawal

Name of Candidate: Isha Agrawal

University Roll No: 201500304

Sign: Madhav Maheshwari

Name of Candidate: Madhav Maheshwari

University Roll No: 201500378

ABSTRACT

An abstract of a food delivery web application would describe the fundamental features of the platform. A food delivery web application would enable users to order food from nearby restaurants and have it delivered to their location. The application would have a user-friendly interface, allowing users to browse restaurants and menus, select items, and place orders. It would also provide users with real-time updates on the status of their order and the estimated delivery time. The application would allow users to make payments online, providing a secure and convenient way to complete transactions. Additionally, the platform would offer customer support to address any issues that may arise during the ordering and delivery process. Overall, a food delivery web application would provide a convenient and efficient way for users to order food from their favourite local restaurants.

An abstract of another food delivery web application could focus on different features and aspects of the platform. For example, the application could offer a personalized user experience, using artificial intelligence and machine learning algorithms to suggest dishes and restaurants based on the user's preferences and past orders. The platform could also allow users to track the delivery driver's location in real-time and communicate with them directly through the app. The application could offer a subscription-based model for frequent users, providing discounts and other benefits. The platform could also have a rating and review system for restaurants and drivers, allowing users to provide feedback and help improve the overall experience. Finally, the application could integrate with other food-related services, such as grocery delivery and meal kit delivery, offering a more comprehensive solution for users' food needs.

CONTENT

Chapter: 1

Introduction

○ Overview

A food delivery web application is an online platform that allows users to order food from local restaurants and have it delivered to their location. The platform typically features a user-friendly interface that allows customers to browse restaurant menus, select items, and place orders. The application may also provide users with real-time updates on the status of their order and the estimated delivery time.

Food delivery web applications usually partner with a network of restaurants in the local area and enable them to receive orders directly from the platform. The application may offer different delivery options, such as standard delivery or express delivery, and may provide users with the ability to track the delivery driver's location and estimated time of arrival.

In addition, food delivery web applications may offer payment processing capabilities, allowing users to pay for their orders online using credit or debit cards, digital wallets, or other payment methods. The platform may also provide customer support to address any issues that arise during the ordering and delivery process.

Overall, a food delivery web application provides a convenient and efficient way for users to order food from their favorite local restaurants without having to leave their homes or offices. The platform benefits both users and restaurants, providing an additional revenue stream for restaurants while offering users a wide selection of food options and a convenient delivery service.

○ Background Study

The concept of food delivery has been around for many decades, with delivery services being offered by various restaurants and fast-food chains. However, the rise of the internet and mobile devices has led to the development of food delivery web applications, making it easier and more convenient for customers to order food from their favourite restaurants.
The first food delivery web application was launched in 1995, with the founding of the website OrderPizza.com. This website allowed customers to order pizza online from various pizza chains, including Pizza Hut and Domino's, and have it delivered to their location. Since then, the food delivery industry has grown rapidly, with the development of various food delivery web applications such as Grub hub, Door Dash, Uber Eats, and Postmates.
These web applications operate on a commission-based model, where they partner with local restaurants to offer food delivery services to customers. The platform takes a percentage of the restaurant's sales, typically ranging from 15% to 30%, as a commission fee for using their platform. In return, the platform provides restaurants with a new revenue stream and exposure to a wider audience.

○ Project Planning

Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment. Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure. The logical dependencies between tasks are defined using an activity network diagram that enables identification of the critical path. Float or slack time in the schedule can be calculated using project management software. Then the necessary resources can be estimated and costs for each activity can be allocated to each resource, giving the total project cost. At this stage, the project plan may be optimized to achieve the appropriate balance between resource usage and project duration to comply with the project objectives. Once established and agreed, the plan becomes what is known as the baseline. Progress will be measured against the baseline throughout the life of the project.

Purposes

The purposes of AI image generator technology are varied and depend on the specific The main purpose of a food delivery web application is to provide a convenient and efficient way for users to order food from local restaurants and have it delivered to their location. However, there are several other purposes of food delivery web applications, including:

Increased revenue for restaurants: By partnering with food delivery web applications, restaurants can reach a wider audience and increase their sales revenue. They can also save costs on delivery infrastructure, such as drivers and vehicles, as the platform takes care of the delivery process.

Convenience for users: Food delivery web applications offer users the convenience of ordering food from their favorite restaurants without having to leave their homes or offices. Users can easily browse restaurant menus, select items, and place orders using a simple and user-friendly interface.

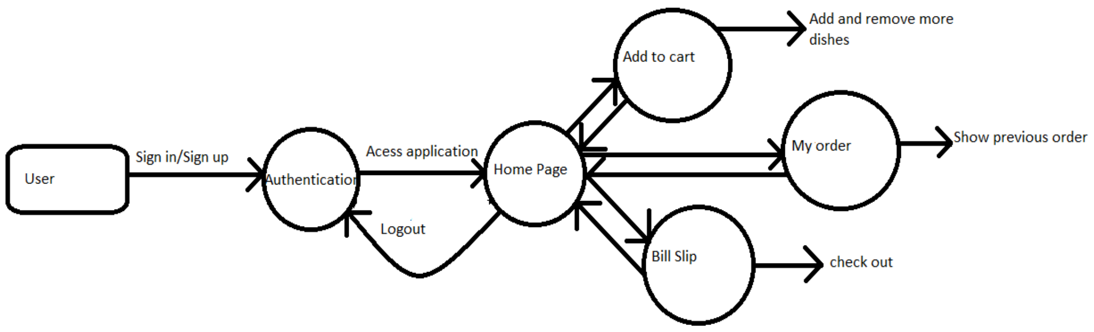
Variety of food options: Food delivery web applications offer a wide selection of food options, allowing users to choose from different cuisines and types of restaurants. This variety provides users with more choices and the ability to try new dishes and restaurants.

○

Chapter: 2

System Design

Design



○

Chapter: 3

Hardware and Software Requirements

○ Hardware Requirement

Processor	: Minimum Dual Core
Operating System	: Windows
Ram	: Minimum 512mb
Hardware device	: Mobile or Computer
Storage	: Minimum 4Gb
Display	: Any Display

○ Software Requirement

Front-end Technologies	: React JS
Language Used	: JavaScript
Back-End	: Open AI, Mongo DB, Express, Node JS
Database	: Mongo DB
Web Browser	: Chrome, Firebox, Microsoft Edge

Chapter: 4

Implementing Tools for the Project

Tools

- React JS
- JavaScript
- Mongo DB
- Node JS
- Bcrypt
- JWT Token
- Express

React JS

React framework is an open-source JavaScript framework and library developed by Facebook, it’s used for building interactive used interface and web application quickly and efficiently with significantly less code than you would with vanilla java scripts.

JavaScript

JavaScript is a dynamic programming language that's used for web development, in web applications, for game development, and lots more. It allows you to implement dynamic features on web pages that cannot be done with only HTML and CSS

Mongo DB

Mongo db. is an open-source NoSQL database management program. NoSQL as a alternative to traditional relational database. NoSQL database are quite useful for working with large sets if distributed data. Mongo DB is a tool can manage document-oriented information, store are retrieving information.

Node JS

Node JS is an open-source, cross-platform JavaScript runtime environment and library for running web application outside the clint's browser. Node is used extensively for server-side programming, making it possible for developers to use java scripts for client side and server side code without needing to learn an additional language.

Bcrypt

Bcrypt is a password hashing algorithm designed by Niels Provos and David Mazières based on the Blowfish cipher. The name “bcrypt” is made of two parts: b and crypt, where b stands for Blowfish and crypt is the name of the hashing function used by the Unix password system

JWT token

JSON web token (JWT), pronounced "jot", is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object

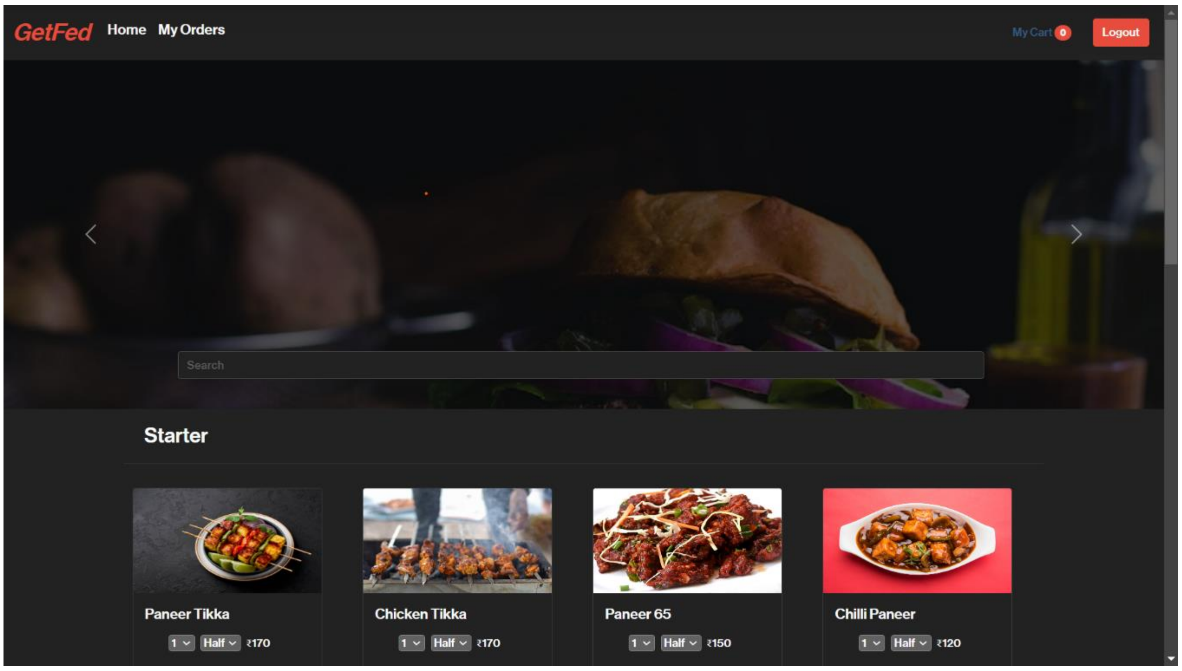
Express

Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application. It's a layer built on the top of the Node js that helps manage servers and routes

Chapter: 5

Project View Model

Home Page

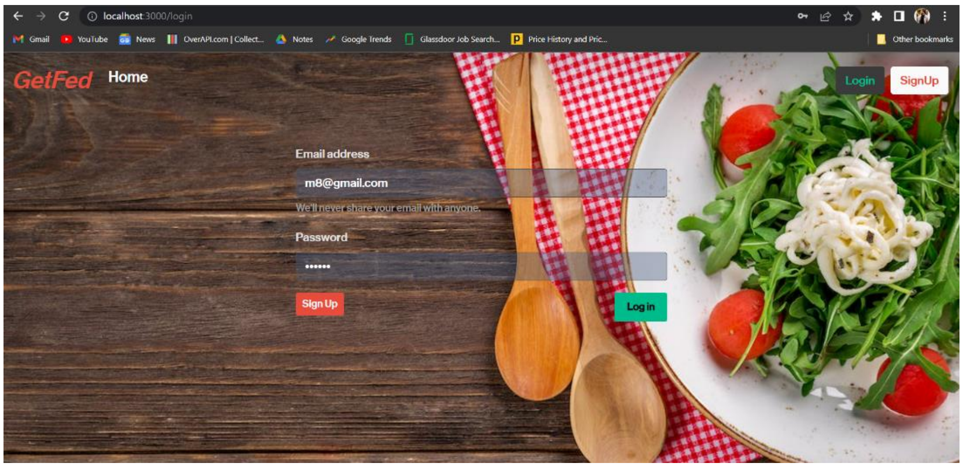


```
import React, { useEffect, useState } from "react";
import Navbar from "../Components/Navbar";
import Footer from "../Components/Footer";
import Card from "../Components/Card";
// import Carousal from "../Components/Carousal";

export default function Home() {
  const [search,setSearch]=useState('');
  const [foodCat, setFoodCat] = useState([]);
  const [foodItem, setFoodItem] = useState([]);

  const loadData = async () => {
    let response = await fetch("http://localhost:5000/api/foodData", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
    });
    response = await response.json();
    setFoodItem(response[0]);
    setFoodCat(response[1]);
    // console.log(response[0],response[1]);
  };
  useEffect(() => {
    loadData();
  }, []);

  return (
    <div>
      <div>
        <Navbar />
      </div>
```

```
<div>
  <div
    id="carouselExampleFade"
    className="carousel slide carousel-fade"
    data-bs-ride="carousel"
    style={{ objectFit: "contain !important" }}
  >
    <div className="carousel-inner" id="carousal">
      <div className="carousel-caption" style={{ zIndex: "10" }}>
        <div className="d-flex justify-content-center">
          <input
            className="form-control me-2"
            type="search"
            placeholder="Search"
            aria-label="Search"
            value={search}
            onChange={(e)=>{setSearch(e.target.value)}}
          />
          {/* <button
            className="btn btn-outline-success text-white bg-success"
            type="submit"
          >
            Search
          </button> */}
        </div>
      </div>
      <div className="carousel-item active">
        <img
          src="https://source.unsplash.com/random/900x700/?burger"
          className="d-block w-100"
          style={{ filter: "brightness(40%)" }}
          alt="..."
        />
      </div>
      <div className="carousel-item">
        <img
          src="https://source.unsplash.com/random/900x700/?pastry"
          className="d-block w-100"
          style={{ filter: "brightness(40%)" }}
          alt="..."
        />
      </div>
      <div className="carousel-item">
        <img
          src="https://source.unsplash.com/random/900x700/?barbeque"
          className="d-block w-100"
          style={{ filter: "brightness(40%)" }}
          alt="..."
        />
      </div>
    </div>
    <button
      className="carousel-control-prev"
      type="button"
      data-bs-target="#carouselExampleFade"
      data-bs-slide="prev"
    >
      <span
        className="carousel-control-prev-icon"
        aria-hidden="true"
      ></span>
      <span className="visually-hidden">Previous</span>
    </button>
```

```
          <button
            className="carousel-control-next"
            type="button"
            data-bs-target="#carouselExampleFade"
            data-bs-slide="next"
          >
            <span
              className="carousel-control-next-icon"
              aria-hidden="true"
            ></span>
            <span className="visually-hidden">Next</span>
          </button>
        </div>
      </div>
      <div className="container text-white">
        {foodCat !== []
          ? foodCat.map((data) => {
              return (
                <div className="row mb-3">
                  <div key={data._id} className="fs-3 m-3">
                    {data.CategoryName}
                  </div>
                  <hr />
                  {foodItem !== [] ? (
                    foodItem
                      .filter((item)      =>      (item.CategoryName      ===      data.CategoryName)      &&
(item.name.toLowerCase().includes(search.toLocaleLowerCase())))
                      .map((filterItems) => {
                        return (
                          <div
                            key={filterItems._id}
                            className="col-12 col-md-6 col-lg-3"
                          >
                            <Card
                              foodItem={filterItems}
                              options={filterItems.options[0]}

                            ></Card>
                          </div>
                        );
                      })
                  ) : (
                    <div>No such Data Found</div>
                  )}
                </div>
              );
            })
          : ""}
      </div>
      <div>
        <Footer />
      </div>
    </div>
  );
}


//console.log(process)
//(process.argv)
```

Login

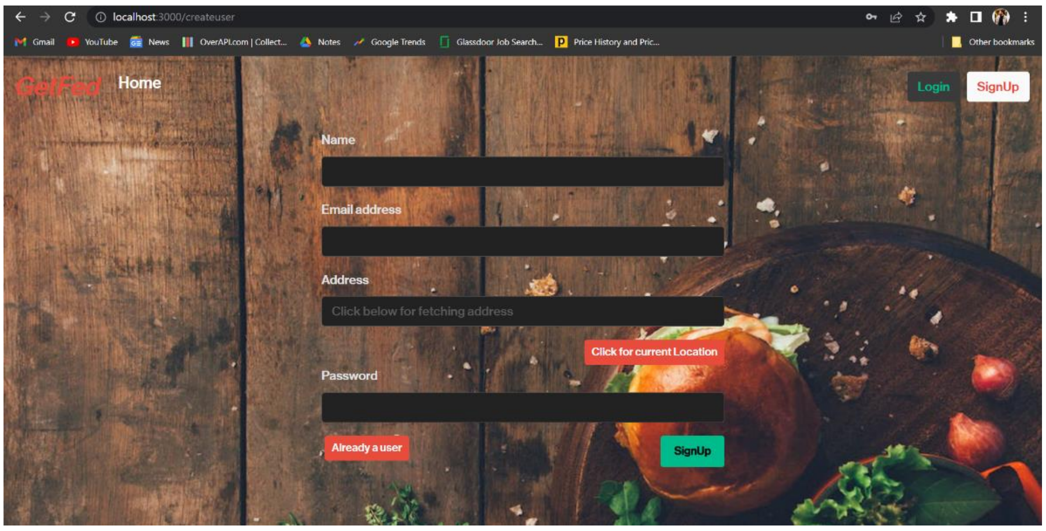


Email address
m8@gmail.com
We'll never share your email with anyone.
Password
••••••

```
import React, { useState } from 'react'
import Navbar from '../Components/Navbar';
import { useNavigate, Link } from 'react-router-dom'
export default function Login() {
  const [credentials, setCredentials] = useState({ email: "", password: "" })
  let navigate = useNavigate()

  const handleSubmit = async (e) => {
    e.preventDefault();
    const response = await fetch("http://localhost:5000/api/loginuser", {
      // credentials: 'include',
      // Origin:"http://localhost:3000/login",
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({ email: credentials.email, password: credentials.password })

    });
    const json = await response.json()
    console.log(json);
    if (json.success) {
      //save the auth toke to local storage and redirect
      localStorage.setItem('userEmail', credentials.email)
      localStorage.setItem('authToken', json.authToken)
      navigate("/");

    }
    else {
      alert("Enter Valid Credentials")
    }
  }

  const onChange = (e) => {
    setCredentials({ ...credentials, [e.target.name]: e.target.value })
  }

  return (
    <div style={{backgroundImage: 'url("https://images.pexels.com/photos/326278/pexels-photo-326278.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1")', height: '100vh', backgroundSize: 'cover' }}>
      <div>
        <Navbar />
      </div>
      <div className='container'>
        <form className='w-50 m-auto mt-5 rounded' onSubmit={handleSubmit}>
```

```
          <div className="m-3">
            <label htmlFor="exampleInputEmail1" className="form-label">Email address</label>
            <input type="email" className="form-control" name='email' value={credentials.email}
onChange={onChange} aria-describedby="emailHelp" />
            <div id="emailHelp" className="form-text">We'll never share your email with
anyone.</div>
          </div>
          <div className="m-3">
            <label htmlFor="exampleInputPassword1" className="form-label">Password</label>
            <input type="password" className="form-control" value={credentials.password}
onChange={onChange} name='password' />
          </div>
          <div className='m-3'>

          <button type="submit" style={{fontWeight:'bold'}} className="btn btn-success btn-sm
float-end p-2 px-3">Log in</button>
          <Link to="/createuser" className="btn btn-danger btn-sm float-start">Sign Up</Link>
          </div>
        </form>

      </div>
    </div>
  )
}
```

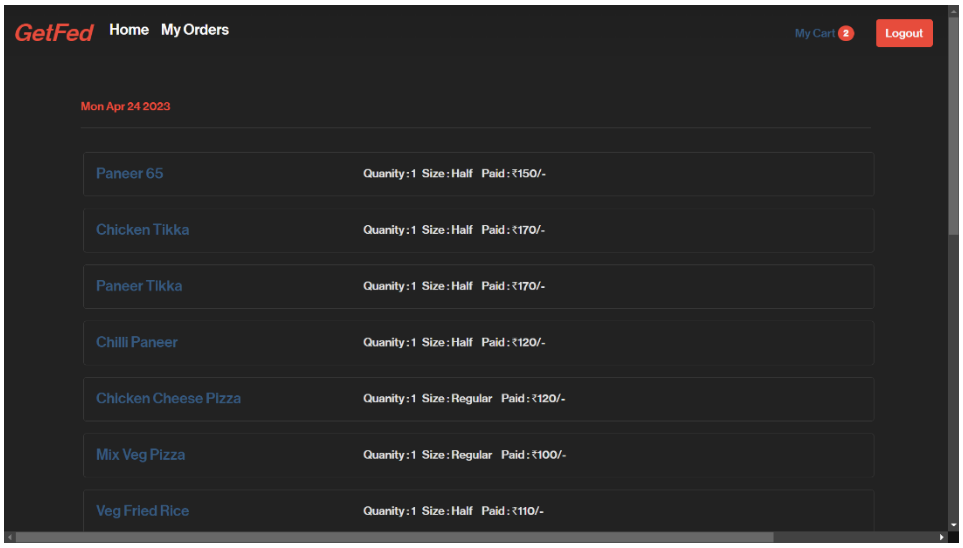
Sign Up


localhost:3000/createuser
GetFed
Home
Login
SignUp
Name
Email address
Address
Click below for fetching address
Click for current Location
Password
Already a user
SignUp

```
import React, { useState } from 'react'
import { useNavigate, Link } from 'react-router-dom'
import Navbar from '../Components/Navbar';

export default function Signup(e) {
  const [credentials, setCredentials] = useState({ name: "", email: "", password: "", geolocation: ""
})
  let [geolocation, setAddress] = useState("");
  let navigate = useNavigate()

  const handleClick = async (e) => {
    e.preventDefault();
    let navLocation = () => {
      return new Promise((res, rej) => {
        navigator.geolocation.getCurrentPosition(res, rej);
      });
    }
    let latlong = await navLocation().then(res => {
      let latitude = res.coords.latitude;
      let longitude = res.coords.longitude;
      return [latitude, longitude]
    })
    // console.log(latlong)
    let [lat, long] = latlong
    console.log(lat, long)
    const response = await fetch("http://localhost:5000/api/getlocation", {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({ latlong: { lat, long } })

    });
    const { location } = await response.json()
    // console.log(location);
    setAddress(location);
    setCredentials({ ...credentials, [e.target.name]: location })
  }

  const handleSubmit = async (e) => {
    e.preventDefault();
    const response = await fetch("http://localhost:5000/api/createuser", {
      // credentials: 'include',
      // Origin:"http://localhost:3000/login",
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({ name: credentials.name, email: credentials.email, password:
credentials.password, location: credentials.geolocation })

    });
    const json = await response.json()
    console.log(json);
    if (json.success) {
      //save the auth toke to local storage and redirect
      // localStorage.setItem('authToken', json.authToken)
      navigate("/login")

    }
    else {
      alert("Enter Valid Credentials")
    }
  }

  const onChange = (e) => {
    setCredentials({ ...credentials, [e.target.name]: e.target.value })
  }
  const twoF=(e)=>{
    setAddress(e.target.value)
    onChange(e);
  }

  return (
    <div style={{ backgroundImage: 'url("https://images.pexels.com/photos/1565982/pexels-photo-
1565982.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=1")', backgroundSize:
'cover',height: '100vh' }}>
```

```
   <div>
   <Navbar />
   </div>

     <div className='container' >
       <form className='w-50 m-auto' onSubmit={handleSubmit}>
         <div className="m-3">
           <label htmlFor="name" className="form-label">Name</label>
           <input type="text" className="form-control" name='name' value={credentials.name}
onChange={onChange} aria-describedby="emailHelp" />
         </div>
         <div className="m-3">
           <label htmlFor="email" className="form-label">Email address</label>
           <input type="email" className="form-control" name='email' value={credentials.email}
onChange={onChange} aria-describedby="emailHelp" />
         </div>

         <div className="m-3">
           <label htmlFor="geolocation" className="form-label">Address</label>
           <fieldset>
             <input type="text" className="form-control" name='geolocation' placeholder="Click
below for fetching address" value={geolocation} onChange={twoF} aria-
describedby="emailHelp" />
           </fieldset>
         </div>
         <div className="m-3">
           <button type="button" onClick={handleClick} name="geolocation" className=" btn
btn-danger btn-sm float-end">Click for current Location </button>
         </div>


         <div className="m-3 mt-5">
           <label htmlFor="exampleInputPassword1" className="form-label">Password</label>
           <input type="password" className="form-control" value={credentials.password}
onChange={onChange} name='password' />
         </div>
         <div className='m-3'>
         <button type="submit" style={{fontWeight:'bold'}} className="btn btn-success btn-sm
float-end p-2 px-3">SignUp</button>
           <Link to="/login" className="mx-1 btn btn-danger btn-sm float-start">Already a
user</Link>
         </div>
       </form>
     </div>
   </div>
```

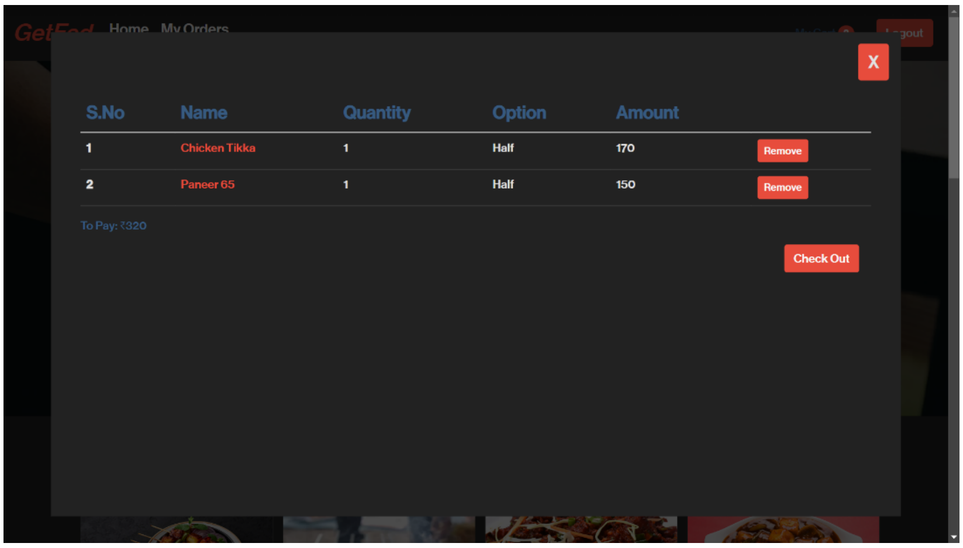
My Order



Mon Apr 24 2023

Paneer 65	Quanity : 1 Size : Half Paid : ₹150/-
Chicken Tikka	Quanity : 1 Size : Half Paid : ₹170/-
Paneer Tikka	Quanity : 1 Size : Half Paid : ₹170/-
Chilli Paneer	Quanity : 1 Size : Half Paid : ₹120/-
Chicken Cheese Pizza	Quanity : 1 Size : Regular Paid : ₹120/-
Mix Veg Pizza	Quanity : 1 Size : Regular Paid : ₹100/-
Veg Fried Rice	Quanity : 1 Size : Half Paid : ₹110/-

Cart



S.No	Name	Quantity	Option	Amount	
1	Chicken Tikka	1	Half	170	Remove
2	Paneer 65	1	Half	150	Remove

To Pay: ₹320

```
import React from 'react'
import {useCart ,useDispatchCart} from '../Components/ContextReducer'
// import Delete from '@material-ui/icons/Delete'

export default function Cart() {
  let data =useCart()
  let dispatch = useDispatchCart();
  if(data.length===0){
    return (
      <div>
      <div className="m-5 w-100 text-center fs-3">The Cart is Empty!</div>
      </div>
    )
  }
  const handleCheckOut=async()=>{
    let userEmail=localStorage.getItem('userEmail')
    let response=await fetch('http://localhost:5000/api/orderData',{
    method:'POST',
    headers:{
      'Content-Type':'application/json'
    },
    body:JSON.stringify({
      order_data:data,
      email:userEmail,
      order_date:new Date().toDateString()
    })
    }
    );
    console.log("Order",response)
    if(response.status===200){
      dispatch({type:'DROP'})
    }
  }
  let totalPrice=data.reduce((total,food)=>total+food.price,0)
  return (
    <div>
      <div className='container m-auto table-responsive table-responsive-sm table-responsive-md
overlowAuto my-3'>
      <table className='table table-hover'>
        <thead className='text-primary fs-4'>
          <tr>
            <th scope='col'>S.No</th>
            <th scope='col'>Name</th>
            <th scope='col'>Quantity</th>
            <th scope='col'>Option</th>
            <th scope='col'>Amount</th>
            <th scope='col'></th>
          </tr>
        </thead>
        <tbody>
          {data.map((food,index)=>(
            <tr>
            <th scope='row'>{index+1}</th>
            <td className='text-danger'>{food.name}</td>
            <td>{food.qty}</td>
            <td className='text-capitalize'>{food.size}</td>
            <td>{food.price}</td>
            <td ><button type="button" className="btn p-0">
            <button type="button" class="btn btn-danger btn-sm" onClick={() => { dispatch({ type:
"REMOVE", index: index }) }}>Remove</button>
            </button> </td>
            </tr>
          ))}
        </tbody>
      </table>
      <div>
        <h1 className='fs-6 text-primary'>To Pay: ₹{totalPrice}</h1>
      </div>
<div className='m-3'>
  <button className='btn btn-danger float-end' onClick={handleCheckOut}>Check Out</button>
</div>
      </div>
    </div>
```

○

Chapter: 6

Software Testing

Software testing is the process of evaluating a software program with the aim of finding out whether it meets the specified requirements set out at the planning stage. In other words, software testing ensures that your software does what you want it to do and doesn't do anything you don't want it to do.

○ Why software testing is needed?

Ultimately, the purpose of software testing is to help ensure that a software product operates in line with both business, technical and user requirements. Specifically, its objectives include:
Verifying that the software conforms to the technical specifications set out at the planning stage. This usually involves checking documents, code and designs to ensure the software has been built in line with requirements.
Validating that the software program meets user requirements and has the potential to achieve the desired results for the business.
Finding and preventing defects that may cause the software to crash or fail when going live or that may impede the functionality or reliability of the application.
Gathering information about the software, including any defects or bugs it has. This data can be used to prevent and fix future issues and to give stakeholders more insight into the software and its development and performance.
Ensuring compatibility with different operating systems and device types.
Ensuring optimal user experience through rigorous checks to find out how easy the software is to use and whether it delivers an enjoyable experience.

○ Testing Strategy

There are types of testing that we implement. They are as follows:

While deciding on the focus of testing activities, study project priorities. For example, for an online system, pay more attention to response time. Spend more time on the features used frequently. Decide on the effort required for testing based on the usage of the system. If the system is to be used by a large number of users, evaluate the impact on users due to a system failure before deciding on the effort.

This create two problem

o

- Time delay between the cause and appearance of the problem.
- The effect of the system errors on files and records within the system.
-

The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the systems to limits. The testing process focuses on the logical intervals of the software ensuring that all statements have been tested and on functional interval is conducting tests to uncover

errors and ensure that defined input will produce actual results that agree with the required results. Program level testing, modules level testing integrated and carried out.

There are two major types of testing they are:

- White Box Testing
- Black Box Testing

o White Box Testing

White box sometimes called “Glass box testing” is a test case design uses the control structure of the procedural design to drive test case. Using white box testing methods, the following tests where made on the system

a) All independent paths within a module have been exercised once. In our system, ensuring that case was selected and executed checked all case structures. The bugs that were prevailing in some part of the code where fixed
b) All logical decisions were checked for the truth and falsity of the values.

o Black Box Testing

Black box testing focuses on the functional requirements of the software. This is black box testing enables the software engineering to derive a set of input conditions that will fully exercise all functional requirements for a program. Black box testing is not an alternative to white box testing rather it is complementary approach that is likely to uncover a different class of errors that white box methods like.

- Interface errors.
- Performance in data structure.
- Performance errors.
- Initializing and termination errors.

Chapter: 7
Conclusion

○ Conclusion

In conclusion, a food delivery web app is a useful and convenient tool for customers who want to order food online and have it delivered to their doorstep. The benefits of a food delivery web app include easy access to menus, the ability to customize orders, and convenient payment options.

Food delivery web apps have also become an important part of the restaurant industry, providing a platform for restaurants to reach new customers and increase their revenue. The features of a food delivery web app can include real-time order tracking, push notifications, and customer loyalty programs.

However, food delivery web apps also have some limitations, such as delivery fees and minimum order requirements. There is also a concern for the safety and well-being of delivery drivers, who are often underpaid and overworked.

Overall, a food delivery web app can be a valuable tool for both customers and restaurants, but it is important to consider the potential drawbacks and take steps to address any issues that arise.

○ Future Aspect

The future of food delivery web apps looks promising, as technology continues to advance and new trends emerge in the food industry. Here are some potential future aspects of food delivery web apps:

Improved personalization: With the help of artificial intelligence (AI), food delivery web apps can provide more personalized recommendations to customers based on their preferences and past orders. This can lead to increased customer satisfaction and loyalty.

Integration with smart home devices: Food delivery web apps can integrate with smart home devices like Amazon Alexa or Google Home, allowing customers to place orders with voice commands and track their deliveries in real-time.

Expansion to new markets: As food delivery services continue to gain popularity, we can expect to see more food delivery web apps expanding to new markets and offering more diverse food options.