

**C.K. Pithawalla College of Commerce-Management-Computer Application, Surat,Gujarat**

VEER NARMAD SOUTH GUJARAT UNIVERSITY, SURAT

Project Report On

**Food Delivery Website**

AS A PARTIAL REQUIREMENT FOR THE DEGREE OF

BACHELOR OF COMPUTER APPLICATION

(B.C.A)

**2024-2025**

GUIDED BY : SUBMITTED BY :

**Prof. Hetal Mehata** **VISHWAKARMA ABHISHEK RAMVRIKSH**

**DUBEY VISHAL JAYKUMAR**

**SINGH ABHISHEK JITENDRA**

**CHAUDHARI AJIT ASHOKKUMAR**

**Acknowledgement**

The reason of completing the project work successfully is not just our efforts but efforts of many people. The people, who trusted, guided and encouraged us with every means. Guide is a person who provides you the direction towards success, so we feel great pleasure to express our gratitude to all our guides, our faculty members as well as every person who helped me directly or indirectly with our project.

We would like to thank **Dr. Marteen Patel** (In-charge Principal at C.K.Pithawalla College)and **Mr. Gaurang Joshi** (BCA-HOD at C.K.Pithawalla College)for granting us an opportunity to work on this project. Their skills and experience was a guiding path in this learning process. She made constant efforts to shape up our skills as per the industry Standards. She provided us a very homely and friendly environment which made it the best place to work. Their guidance was really priceless and will always be a guiding light in Industry.

We are also indebted to our Professor **Mr. Hetal Mehta** who provided constant encouragement, support & valuable guidance before and during our project. It was her effort who led us to this place for project work. Her guidance and suggestions were invaluable.

Thank you very much

**VISHWAKARMA ABHISHEK RAMVRIKSH**

**DUBEY VISHAL JAYKUMAR**

**SINGH ABHISHEK JITENDRA**

**CHAUDHARY AJIT ASHOKKUMAR**

|  |  |  |  |
| --- | --- | --- | --- |
| Index | | | |
| SrNo | **Chapter** | | **Page No** |
| 1 | **Introduction** | | **1** |
|  | 1.1 | Project Profile | 2 |
|  | 1.2 | Project Introduction | 3 |
| 2 | **Environment Description** | | **4** |
|  | 2.1 | Hardware Software Requirement | 5 |
|  | 2.2 | Tools And Technology | 6 |
| 3 | **Existing System** | | **10** |
|  | 3.1 Introduction | | 11 |
|  | 3.2 Limitation | | 12 |
| 4 | **Proposed System** | | **13** |
|  | 4.1 | Scope | 14 |
|  | 4.2 | Aim And Objective | 18 |
|  | 4.3 | Excepted Advantage | 20 |
| 5 | **System Planning** | | **23** |
|  | 5.1 | Requirement Specification | 24 |
|  | 5.2 | Feasibility Study (technical, Economy, and operational) | 28 |
|  | 5.3 | Life Cycle Model | 29 |
|  | 5.4 | Effort Distribution | 30 |
|  | 5.5 | Task Dependency Diagram | 31 |
|  | 5.6 | Project Schedule Chart (Time Line Chart ) | 32 |
| 6 | **System Model Architecture** | | **33** |
|  | 6.1 | Use Case Diagram | 34 |
|  | 6.2 | Activity Diagram | 35 |
| 7 | **System Diagram** | | **37** |
|  | 7.1 | Navigation Map (Site Map Diagram) | 38 |
|  | 7.2 | Database Diagram (Table Relationship Diagram) | 40 |
|  | 7.3 | Table Structure | 42 |
|  | 7.4 | Screen Layout | 45 |
| 8 | **System Testing** | | **50** |
|  | 8.1 | Testing Introduction | 51 |
|  | 8.2 | Test Case | 52 |
| 9 | **System Limitation And Future Enhancement** | | **53** |
| 10 | **References** | | **57** |
|  | 10.1 | Bibliography | 58 |
|  | 10.2 | Webography | 59 |

**1**

Chapter

**Introduction**

|  |  |
| --- | --- |
| 1.1 | Project Profile |
| 1.2 | Project Introduction |

**1.1 Project Profile**

|  |  |
| --- | --- |
| Project Title : | Food Delivery Website |
| Objective : | To provide platform for purchasing food & Delivery |
| Name of the institute : | C.K. Pithawala College Of Commerce Management & Computer Application |
| Developed For : | To provide platform for purchasing food & Delivery |
| Project Guide : | Mr. Hetal Mehta |
| Front End : | HTML, CSS, JavaScript, React.js |
| Back End : | Node.js, Express.js |
| Team Members : | 04 |

**1.2 Project Profile**

* **Seamless Order Management** – Automates and simplifies order placement, tracking, and delivery processes for a hassle-free experience.
* **User-Friendly Interface** – Provides an intuitive and easy-to-navigate platform for customers to browse menus, place orders, and make payments effortlessly.
* **Real-Time Order Tracking** – Enables customers to track their orders in real time, ensuring transparency and reducing uncertainty.
* **Multiple Payment Options** – Supports various payment methods, including credit/debit cards, UPI, digital wallets, and cash on delivery for user convenience.
* **Efficient Delivery Management** – Streamlines delivery processes by assigning orders to nearby delivery agents for faster and accurate deliveries.
* **Comprehensive Restaurant Management** – Allows restaurants to manage menus, prices, discounts, and availability dynamically.
* **Customer Reviews & Ratings** – Enables users to rate and review restaurants and delivery services to maintain quality and trust.
* **Loyalty Programs & Discounts** – Offers promotional deals, discount codes, and loyalty rewards to enhance customer engagement and retention.
* **Robust Admin Dashboard** – Provides an advanced dashboard for managing orders, tracking revenue, monitoring deliveries, and handling customer support.
* **Scalable & Mobile-Optimized** – Designed to handle high traffic and function smoothly across all devices, including mobile phones and tablets.

**2**

Chapter

**Environment Description**

|  |  |
| --- | --- |
| 2.1 : | Hardware and Software Requirements |
|  | 2.1.1 Development Tools |
|  | 2.1.2 Client Side Tool |
|  | 2.1.3 Server Side Tool |
| 2.2 : | **Tools and Technology** |
|  | 2.2.1 Core Technology |
|  | 2.2.2 Extra Tools |
|  |  |

2.1 Hardware And Software Requirements

* Client Side :
* **Frontend Frameworks** – React.js (for better SEO and performance)
* **Styling** – CSS
* **State Management** – Redux, Context API
* **JavaScript Runtime** – Node.js (for SSR with Next.js)
* **Libraries for UI Enhancements** – Framer Motion (for animations)
* Server Side :
* **Backend Framework** – Express.js (Node.js)
* **Database** – MongoDB with Mongoose
* **Authentication** – JWT (JSON Web Tokens)
* **Payment Integration** – Stripe
* **Cloud Storage** – AWS S3 (MongoDB Cluster)
* Development Side :
* **Code Editor/IDE** – VS Code
* **Version Control System** – Git, GitHub
* **Database Management System** – MongoDB,
* **API Testing Tool** – Postman

2.2 Tools And Technology

**Technology:**

* **Core Technology** :
* **MongoDB :**

**- A NoSQL database that stores data in JSON-like documents for flexibility and scalability.**

* **Express.js :**

**- A lightweight web framework for Node.js that simplifies backend development.**

* **React.js :**

**- A frontend library for building dynamic and interactive user interfaces.**

* **Node.js :**

**- A frontend library for building dynamic and interactive user interfaces.**

**How MERN Works? :**

* **Frontend (React.js): Users interact with the UI built using React.**
* **Backend (Express.js & Node.js): Handles API requests, business logic, and database interactions.**
* **Database (MongoDB): Stores and retrieves data efficiently.**
* **Extra Technology:**

* **Axios:**

**- Axios** is a JavaScript library used for making HTTP requests (API calls) from the frontend to the backend.

**-** Supports **GET, POST, PUT, DELETE** requests

**-** Works in both browser & Node.js environments

- Handles automatic JSON conversion

- Supports **interceptors** for request/response manipulation

* **CSS:**

**-** **CSS** is a stylesheet language used to describe the presentation of HTML or XML documents**.**

**-** **CSS** controls layout, colors, fonts, and overall design, separating content from presentation.

**-** **A CSS** rule consists of a selector and declaration block.

**-A** declaration block contains properties and values

**-** **Control** how elements are placed with static, relative, absolute, fixed, and sticky positioning.

**-** **Enable** animations and smooth style changes using CSS transitions and keyframes.

* **Mongoose:**

**- Mongoose** is an ODM (Object Data Modeling) library for MongoDB in Node.js, which helps interact with the database using schemas and models.

**-** Defines structured schemas for **MongoDB** collections

**-** Provides built-in validation for data consistency

**-** Supports query building and middleware hooks

**Tools:**

* **Documentation Tools:**
* **EDraw:**

**- Edraw** Max is a 2D business technical diagramming software which help create flowcharts, organizational charts, mind map, network diagrams, floor plans, workflow diagrams, business charts, and engineering diagrams.

**-** **Edraw** Max gives users a Visio-like, professional quality diagramming tool.

* **MS Word:**

**- Microsoft Word** or **MS**-**WORD** (often called **Word**) is a Graphical **word** processing program that users can type with.

**-** It is made by the computer company **Microsoft**. Its purpose is to allow users to type and save documents.

* **MS Power Point:**

**- PowerPoint** is a slideshow presentation program that's part of the **Microsoft** office suite of tools.

**-PowerPoint** makes it easy to create, collaborate, and present your ideas in dynamic, visually compelling ways.

* **Development Tools:**

VS Code

* **VS Code:**

**- VS Code** is a lightweight, open-source code editor developed by Microsoft. It is widely used for web development, supporting multiple programming languages and frameworks.

**-** To start, it is a clean, functional, and fast code editor.

**- IntelliSense –** Smart code completion for JavaScript, React, Node.js, and more.

**-** **Built-in Git Integration** – Version control with GitHub, GitLab, and Bitbucket.

- **Debugging Tools** – Powerful debugging for JavaScript, Node.js, and Python.

- **Extensions Marketplace** – Thousands of extensions for added functionality.

- **Integrated Terminal** – Run shell commands directly in the editor.

- **Lightweight & Fast** – Optimized for performance, even with large projects.

- **Themes & Customization** – Customize UI, keybindings, and settings.

**3**

Chapter

**Existing System**

|  |  |
| --- | --- |
| 3.1 : | Introduction |
| 3.2 : | **Limitation** |
|  |  |
|  |  |
|  |  |

3.1 Introduction

A **Food Delivery Website** is an online platform that allows users to browse restaurants, place food orders, and get deliveries to their doorstep. It streamlines the process of ordering food, managing restaurants, and tracking deliveries efficiently.

**Key Features:**

1. **User Registration & Authentication:** Allows customers to sign up, log in, and manage their profiles securely.
2. **Restaurant Listings**: Displays a variety of restaurants with menus, ratings, and reviews.
3. **Order Placement & Tracking:** Customers can place orders and track their delivery in real time.
4. **Cart & Checkout System:** Supports adding items to the cart, applying discounts, and secure payment processing.
5. **Payment Gateway Integration:** Allows payments via **UPI, cards, net banking, and wallets** using Razorpay, Stripe, or PayPal.
6. **Delivery Management:** Assigns orders to delivery personnel and tracks their live location.
7. **Review & Rating System:** Enables customers to rate restaurants and share feedback.

**Benefits:**

* **Convenience:** Allows customers to order food from anywhere, anytime.
* **Time-Saving:** Reduces wait times with pre-ordering and scheduled deliveries.
* **Seamless Payment Processing:** Supports multiple payment options for hassle-free transactions.
* **Real-Time Order Tracking:** Enhances customer experience with live delivery updates.

3.2 Limitations

* **Initial Setup Complexity:** Setting up the platform requires integrating restaurants, payment gateways, and delivery logistics, which can be time-consuming.
* **High Development Cost:** Building and maintaining a full-featured food delivery website requires investment in development, hosting, and security measures.
* **Customization Limitations:** Some platforms may not support unique business models, such as multi-vendor setups or commission-based pricing.
* **Technical Issues:** The system may experience bugs, downtime, or performance issues, impacting order processing and user experience.
* **Data Security Risks**: Storing customer information, payment details, and restaurant data digitally poses security risks without strong encryption and security protocols.
* **Integration Challenges**: Connecting third-party services like payment gateways, delivery tracking APIs, and CRM tools can be complex and may lead to data inconsistencies.
* **Regulatory Changes**: The platform must comply with food safety, tax, and business regulations, requiring regular updates to avoid legal issues.
* **User Training**: Restaurant partners and delivery personnel may require training to manage orders efficiently through the system.
* **Limited Scalability**: If the platform is not built with scalability in mind, handling increased traffic and orders can become difficult as the business grows.
* **Dependence on Internet Connectivity**: The website and mobile app require a stable internet connection; any downtime can impact orders and deliveries.

**4**

Chapter

**Proposed System**

|  |  |
| --- | --- |
|  | 4.1 Scope  4.2 Aim and Objective  4.3 Excepted Advantage |

4.1 Scope

A **Food Delivery Website** provides a seamless platform for customers, restaurants, and delivery personnel to interact efficiently. It ensures smooth ordering, payment processing, and delivery management while enhancing user experience through automation and real-time tracking.

**Admin Side (Website Owner/Manager)**

1. **Dashboard and Analytics**
   * Overview of total orders, revenue, customer activity, and restaurant performance.
   * Access to real-time reports on sales, earnings, and delivery metrics.
2. **Restaurant Management**
   * Add, approve, or remove restaurant listings.
   * Manage restaurant profiles, menus, pricing, and service availability.
   * Monitor restaurant performance through ratings and reviews.
3. **Order Management**
   * Track and manage all orders placed on the platform.
   * Resolve issues related to cancellations, refunds, or order disputes.
4. **Payment and Commission Handling**
   * Manage transactions between customers, restaurants, and delivery partners.
   * Set and track commission rates for restaurant partners.
   * Generate financial reports on revenue, payouts, and pending transactions.
5. **Delivery Management**
   * Assign deliveries to riders based on location and availability.
   * Track delivery personnel in real-time to ensure timely deliveries.
6. **User and Role Management**
   * Control access and permissions for admin users, restaurant owners, and delivery personnel.
   * Manage customer accounts, including profile updates and security settings.
7. **Customer Support & Feedback Management**
   * Address customer complaints, refunds, and order issues.
   * Monitor restaurant and delivery personnel ratings and reviews.
8. **Security and Compliance**
   * Implement data encryption and secure payment methods.
   * Ensure compliance with food safety laws, tax regulations, and business policies.

**Customer Side**

1. **User Registration & Profile Management**
   * Sign up/log in using email, phone number, or social media.
   * Update personal details, addresses, and payment methods.
2. **Restaurant Browsing & Food Ordering**
   * Search restaurants by location, cuisine, ratings, or price range.
   * Browse restaurant menus and customize orders.
   * Add items to the cart, apply discounts, and proceed to checkout.
3. Order Tracking & History
   * View real-time order status updates (preparing, out for delivery, delivered).
   * Track delivery personnel via live location sharing.
   * Access past orders and reorder favorite meals.
4. **Payment Processing**
   * Pay securely using **UPI, credit/debit cards, wallets, or cash on delivery**.
   * Save preferred payment methods for faster checkout.
5. **Ratings & Reviews**
   * Rate restaurants and delivery experiences.
   * Leave feedback to improve service quality.
6. Customer Support
   * Contact customer support for order issues, refunds, or complaints.
   * Access FAQs and automated chatbot support.

**Delivery Personnel Side :**

1. Order Assignment & Navigation
   * Receive order assignments with restaurant and customer locations.
   * Use integrated maps for optimized delivery routes.
2. **Earnings & Payment Tracking**
   * View total earnings, completed deliveries, and pending payouts.
3. **Delivery Status Updates**
   * Mark orders as **picked up, out for delivery, or delivered**.
4. **Support & Communication**
   * Contact customers or restaurants for order clarifications.
   * Report delivery issues to the support team.

**Conclusion**

The **Food Delivery Website** provides a **comprehensive ecosystem** where **admins, customers, restaurants, and delivery personnel** work together to ensure efficient and seamless food delivery services. The system enhances **order accuracy, real-time tracking, business growth, and customer satisfaction.**

4.2 Aim and Objective

**Aim**

The primary aim of a **Food Delivery Website** is to provide a seamless, user-friendly, and efficient platform for customers to browse restaurants, place food orders, and receive timely deliveries while ensuring smooth operations for restaurants and delivery personnel.

**Objectives**

1. Automate and Streamline Order Management
   * To simplify the food ordering process for customers.
   * To ensure quick and accurate order processing for restaurants.
2. **Ensure Seamless Customer Experience**
   * To provide an intuitive and responsive interface for easy navigation.
   * To offer real-time order tracking and notifications.
3. Enhance Restaurant and Menu Management
   * To enable restaurants to add, edit, and manage menus efficiently.
   * To provide restaurant owners with sales insights and analytics.
4. Optimize Delivery Operations
   * To assign orders efficiently to nearby delivery personnel.
   * To provide optimized routes for fast and reliable food delivery.
5. Integrate Secure Payment Methods
   * To support multiple payment options, including **UPI, cards, wallets, and COD**.
   * To ensure secure transactions and prevent fraud.
6. Enable Real-Time Order Tracking
   * To allow customers to track food preparation and delivery status.
   * To update customers with estimated delivery times.
7. **Improve Customer Engagement**
   * To offer loyalty programs, discounts, and personalized recommendations.
   * To provide a rating and review system for restaurants and delivery personnel.
8. Enhance Admin & Reporting Capabilities
   * To provide a **powerful admin dashboard** for managing restaurants, orders, and payments.
   * To generate detailed reports for **sales, revenue, customer trends, and delivery performance**.
9. **Ensure Data Security & Compliance**
   * To protect customer and business data with strong encryption.
   * To comply with legal and food safety regulations.

**Conclusion**

The **Food Delivery Website** aims to bridge the gap between customers, restaurants, and delivery personnel through **automation, real-time tracking, and seamless operations**. Its objectives focus on **enhancing user experience, optimizing delivery processes, ensuring secure transactions, and supporting business scalability**.

4.2 Expected Advantages

**1. Improved Order Accuracy**

* **Automated Order Processing**: Reduces human errors in food orders and billing.
* **Consistent Data Handling**: Ensures uniformity in restaurant menus, pricing, and customer details.

**2. Time and Cost Efficiency**

* **Reduced Manual Work**: Automates restaurant order management and delivery scheduling.
* **Lower Administrative Costs**: Minimizes paperwork and administrative overhead.

**3. Fast and Reliable Food Delivery**

* **Timely Deliveries**: Ensures customers receive their orders on time.
* **Efficient Order Processing**: Optimizes restaurant workflows to speed up food preparation and dispatch.

**4. Enhanced Compliance**

* **Regulatory Adherence**: Helps restaurants meet food safety and tax regulations.
* **Up-to-Date Pricing and Tax Management**: Keeps the system updated with GST, VAT, and local tax laws.

**5. Better Data Security**

* **Secured Customer and Restaurant Data**: Protects sensitive user information with encryption.
* **Fraud Prevention: Restricts unauthorized access through authentication mechanisms.**

**6. Comprehensive Reporting & Analytics**

* **Sales and Revenue Reports**: Provides insights into restaurant earnings and customer preferences.
* **Delivery Performance Analytics**: Helps optimize delivery routes and reduce delays.

**7. Customer Self-Service Features**

* **Order Tracking**: Allows users to track food preparation and delivery status in real-time.
* **Easy Reordering**: Saves customer preferences for quick repeat orders.

**8. Simplified Payment and Refund Processing**

* **Multiple Payment Options**: Supports **UPI, debit/credit cards, net banking, and cash on delivery (COD)**.
* **Seamless Refunds**: Automates refund processes for canceled or incorrect orders.

**9. Scalability for Business Growth**

* **Expandable Platform: Supports an increasing number of restaurants, customers, and orders.**
* **Flexible API Integration**: Easily connects with third-party apps like payment gateways and CRM

**10. Improved Accuracy in Tax and Billing Management**

* **Automated Tax Calculations**: Ensures correct billing with applicable taxes.
* **Error-Free Invoices**: Generates accurate receipts and tax reports for restaurants and customers.

**11. Enhanced Customer Satisfaction**

* **Transparent Pricing and Order Details**: Provides clear pricing and estimated delivery times.
* **Error-Free Orders**: Reduces mistakes in food preparation and delivery.

**12. Efficient Audit and Compliance Tracking**

* **Complete Order History**: Maintains a detailed audit log of all transactions.
* **Easy Financial Audits**: Provides structured data for restaurant owners and accountants.

**Conclusion**

The **Food Delivery Website** offers numerous advantages, including **automated order processing, secure payments, enhanced customer satisfaction, scalability, and compliance with regulations**. With streamlined operations for customers, restaurants, and delivery partners, the system ensures a smooth, fast, and error-free food ordering experience.

**5**

Chapter

**System Planning**

|  |  |
| --- | --- |
| 5.1 | Requirement Specification |
| 5.2 | Feasibility Study |
| 5.3 | Life Cycle Model |
| 5.4 | Effort Distribution Diagram |
| 5.5 | Task Dependency Diagram |
| 5.6 | Project Schedule Chart |

5.1 **Requirement Specification**

Here’s a detailed outline for the **requirement specification** of a Payroll Management System:

**1. Functional Requirements**

**1.1 User Management**

* **Customer Registration & Login**: Sign up using email, phone number, or social media.
* **User Profiles**: Manage personal details, delivery addresses, and order history.
* **Role-Based Access**: Different access levels for customers, restaurants, and admin.

**1.2 Restaurant Management**

* **Restaurant Registration**: Restaurants can sign up and create their menus.
* **Menu Management**: Add/edit dishes with names, prices, descriptions, and images.
* **Availability Status**: Restaurants can mark dishes as available or unavailable.

**1.3 Order Management**

* **Browse & Search: Users can search for restaurants, cuisines, and specific dishes.**
* **Cart System**: Add/remove items before checkout.
* **Order Placement**: Secure checkout with multiple payment options.
* **Order Tracking**: Real-time tracking of orders with estimated delivery time.
* **Order History**: View past orders with reorder functionality.

**1.4 Delivery Management**

* **Delivery Boy Panel: Assign and track orders for delivery personnel.**
* **Live GPS Tracking**: Customers can track their delivery agent in real-time.
* **Delivery Notifications**: SMS/email updates for order confirmation, preparation, and delivery.

**1.5 Payment & Billing**

* **Multiple Payment Options**: Credit/debit cards, UPI, wallets, cash on delivery (COD).
* **Invoice Generation: Automatic receipts for every order.**
* **Refund & Cancellation: Users can cancel orders and request refunds if eligible.**

**1.6 Reviews & Ratings**

* **Restaurant & Food Reviews**: Customers can rate and review restaurants and dishes.
* **Delivery Ratings: Users can rate delivery service for feedback improvement.**

**1.7 Offers & Promotions**

* **Discount Coupons**: Apply promo codes at checkout.
* **Loyalty Program**: Earn reward points for future discounts.
* **Restaurant-Specific Deals**: Restaurants can offer discounts on selected items.

**1.8 Admin Dashboard**

* **User & Restaurant Management**: Add, remove, or update users and restaurants.
* **Order & Delivery Monitoring**: Track live orders and delivery status.
* **Financial Reports**: Revenue tracking, commissions, and transaction history.
* **Complaint Handling**: Manage user complaints and refunds.

**2. Non-Functional Requirements**

**2.1 Usability**

* **Simple UI/UX**: Clean, user-friendly interface for customers, restaurants, and delivery partners.
* **Mobile Responsiveness**: Works on desktops, tablets, and smartphones.

**2.2 Performance**

* **Fast Loading**: Quick response times for searches and transactions.
* **High Scalability**: Supports increasing users and restaurant listings.

**2.3 Reliability**

* **Minimal Downtime: Ensure high availability with proper server infrastructure.**
* **Backup System**: Automatic data backups to prevent loss.

**2.4 Security**

* **Secure Transactions**: Encrypted payments with SSL/TLS.
* **User Authentication**: Secure login with OTP or multi-factor authentication.
* **Role-Based Access**: Restrictions for admin, restaurants, and delivery partners.

**2.5 Compliance**

* **Legal Regulations**: Adherence to food safety and data protection laws (GDPR, PCI DSS).
* **Tax Handling**: Automated tax calculations and invoices.

**3. Technical Requirements**

**3.1 Software Requirements**

* **Frontend: React.js / Next.js for a fast and responsive UI.**
* **Backend**: Node.js with Express.js for API handling.
* **Database**: MongoDB for storing user, restaurant, and order data.

**3.2 Hardware Requirements**

* **Cloud Hosting**: AWS / DigitalOcean for scalable hosting.
* **CDN (Content Delivery Network)**: For faster image and menu loading.

**3.3 Third-Party Integrations**

* **Payment Gateway: Stripe, Razorpay, or PayPal integration.**
* **Google Maps API**: For location-based restaurant search and delivery tracking.

**4. Implementation and Support**

**4.1 Deployment**

* **Installation Guide: Step-by-step setup for server and database.**
* **Live Deployment: Smooth transition from testing to production.**

**4.2 Training and Documentation**

* **User Training**: Tutorials for restaurants, delivery agents, and admins.
* **System Documentation: Technical documentation for developers.**

**4.3 Maintenance and Support**

* **24/7 Support**: Technical support for troubleshooting issues.
* **Regular Updates: Feature improvements and security patches.**

**Conclusion**

This **Food Delivery Website** will provide a seamless and secure ordering experience, ensuring **fast delivery, user-friendly navigation, multiple payment options, and real-time tracking**. It caters to **customers, restaurants, and delivery agents**, ensuring smooth operations for all users.

5.2 Feasibility Study

**Feasibility study is an online description of the Website and how it will be used.**

**Three test of feasibility studies by us:**

**1) Technical Feasibility.**

**2) Economical Feasibility.**

**3) Operational feasibility.**

**1) Technical Feasibility**

* **Technical feasibility** assesses the available technology and its compatibility with the website.
* The website will be developed using:
* **Frontend**: React.js (for dynamic UI) and Tailwind CSS (for styling).
* **Backend**: Node.js with Express.js.
* **Database**: MongoDB for scalable and efficient data storage.
* **Server**: Hosted on **AWS / DigitalOcean** for reliability and scalability.
* **Additional Technologies**:
* Google Maps API (for real-time delivery tracking).
* Stripe/Razorpay for secure payments.
* The system will be developed with modern technologies to ensure fast performance, security, and ease of use.

**2) Economical Feasibility**

* **Economic feasibility** examines the financial aspects of the project.
* The project is cost-effective as it primarily requires investment in:
* **Web Hosting & Domain** (minimal cost for initial setup).
* **Third-Party Integrations** (payment gateways, SMS APIs, etc.).
* **Development Costs** (if hiring a development team).
* **Revenue Generation**:
* Commission from restaurants.
* Delivery fees.
* Advertisements and premium listing for restaurants.

**3) Operational Feasibility**

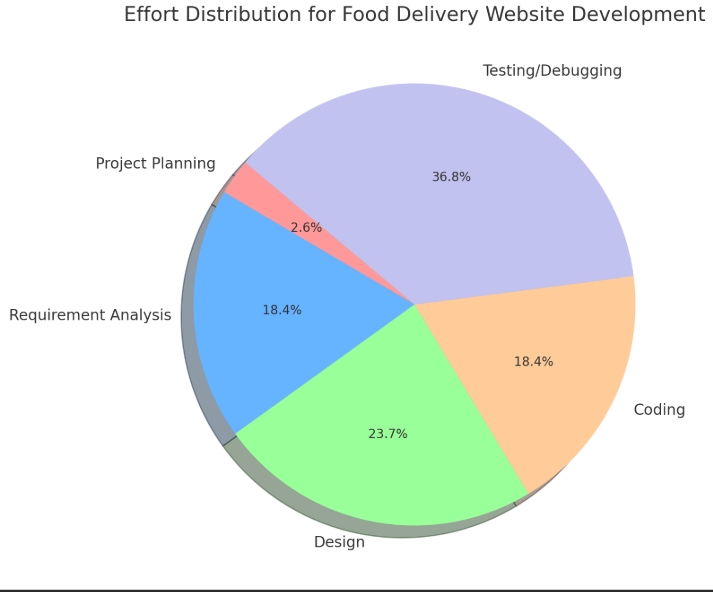
* **Operational feasibility** ensures the system will function efficiently once implemented.
* The system is user-friendly, ensuring smooth navigation for:
* **Customers** (easy food ordering and payment).
* **Restaurants** (efficient order management and menu updates).
* **Delivery Agents** (real-time tracking and notifications).
* The system integrates seamlessly with existing applications like **HR, accounting, and delivery management tools**.
* The platform is **secure, scalable, and reliable**, ensuring **continuous operations without interfering** with other applications.

5.3 Life Cycle Model

* **Increment Model:**
* The **Incremental Model** is an improvement over the **Waterfall Model**, where the software is developed in multiple iterations (increments). Each increment adds new functionality, allowing early delivery of a working system while incorporating customer feedback for future improvements.
* **How It Works**
* The project is divided into small, manageable modules (increments).
* The first increment delivers a **core functional product**, such as basic food ordering.
* After customer feedback, additional features like **real-time tracking, reviews, and promotions** are added in subsequent increments.
* The process continues until the **full system is developed and deployed**.
  + - **Advantages of Incremental Model**
* **Early Delivery of Core Product** – The customer can start using the basic version while further improvements are made.
* **Easy to Identify and Fix Issues** – Since changes are made in smaller iterations, debugging and testing are more manageable.
* **Flexible and Adaptable to Changes** – Customer feedback can be incorporated into future increments without affecting the entire system.
* **Lower Initial Investment** – Since the product is released in phases, businesses can start generating revenue before full development is completed.
* **Better Risk Management** – Problems can be identified at each stage, reducing the risk of project failure.
  + - **Disadvantages**
* **Changing Requirements May Cause Delays** – If business requirements change frequently, previous increments may need to be reworked.

5.4 Effort Distribution Diagram

An **Effort Distribution Diagram** is a graphical representation used to show how effort or resources are distributed across different phases, tasks, or components of a project. It helps in visualizing the allocation of time, effort, or cost, and is useful for project planning, management, and monitoring.



|  |  |
| --- | --- |
| Project Phase | Effort Allocation (%) |
| Project Planning : | 2% - 3% |
| Requirement Analysis : | 10% - 25% |
| Design : | 20% - 25% |
| Coding : | 15% - 20% |
| Testing / Debugging | 30% - 40% |

5.5 Task Dependency Diagram

**Requirement Analysis**

**Planning & Risk Analysis**

**Designing**

**Coding & Integrating Module**

**Testing**

**Requirement Gathering**

**Analyze Gathered Information**

**Determine Scope of the** System

5.6 Project Schedule Chart

* The time line chart describes the time scheduling for project completion
* This time line chart shows project tasks in weekly schedule.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Task** | **Month 1** | | | | | | **Month 2** | | | | **Month 3** | | | |
|  | **W1** | | **W2** | | **W3** | **W4** | **W1** | **W2** | **W3** | **W4** | **W**  **1** | **W**  **2** | **W**  **3** | **W**  **4** |
| 1. **Requirement Gathering** |  | |  |  | |  |  |  |  |  |  |  |  |  |
| Collected Details from developed sites |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Analyzed gathered information |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Determine the scope of the system |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Milestone**:**Req. gathering completed. |  |  | |  | |  |  |  |  |  |  |  |  |  |
| 1. **Planning** |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Basic flow and structure |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Determined different modules |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Milestone: Planning completed |  |  | |  | |  |  |  |  |  |  |  |  |  |
| 1. **Designing** |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Basic Interface design |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Database Design |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Design web forms and modules |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Milestone: Designing Completed |  |  | |  | |  |  |  |  |  |  |  |  |  |
| 1. **Risk Analysis** |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Milestone: Risk Analysis Completed |  |  | |  | |  |  |  |  |  |  |  |  |  |
| 1. **Coding and Integrating modules** |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Implement logic for diff modules |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Integrating code with backend |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Integrating dependent modules |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Milestone: Coding completed |  |  | |  | |  |  |  |  |  |  |  |  |  |
| 1. **Testing of developed system** |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Milestone: Finalized |  |  | |  | |  |  |  |  |  |  |  |  |  |
| 1. **Documentation** |  |  | |  | |  |  |  |  |  |  |  |  |  |
| Milestone: Documentation completed |  |  | |  | |  |  |  |  |  |  |  |  |  |

**6**

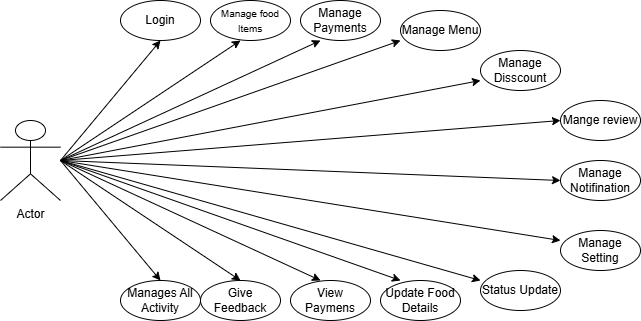
Chapter

**System Model Architecture**

|  |  |
| --- | --- |
| 6.1 | Use Case Diagram |
| 6.2 | Activity Diagram |
|  |  |

6.1 Use Case Digram

* **Admin’s Use Case Diagram:**

****

* **Employee’s Use Case Diagram:**

# 

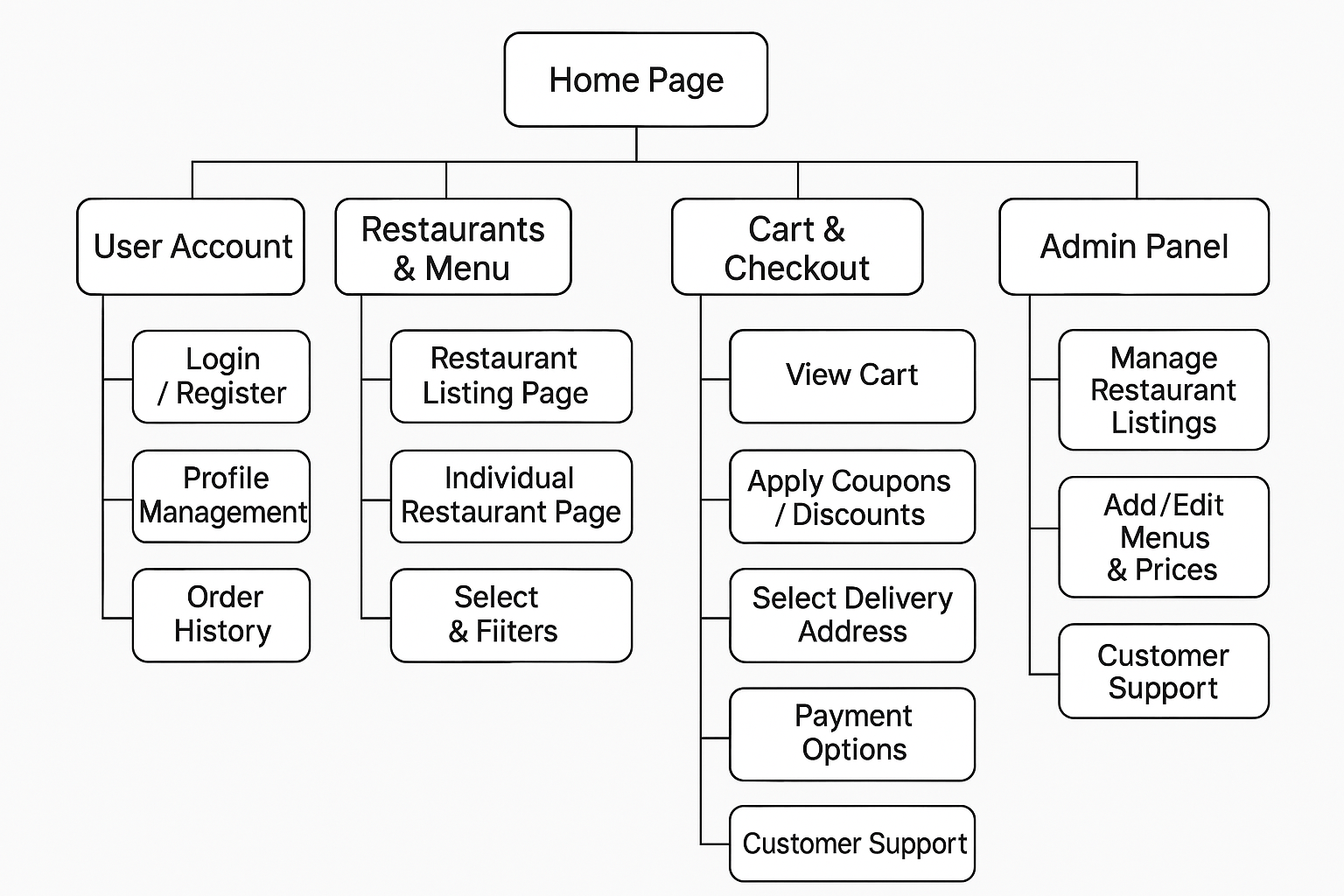
**7**

Chapter

**System Diagram**

|  |  |
| --- | --- |
| 7.1 | Navigation Map |
| 7.2 | Database Diagram |
| 7.3 | Table Structure |
| 7.4 | Screen Layout |

7.1 Navigation Map



**Benefits of a Navigation Map**

1. **Clarity in Structure:**
   * Provides a clear and organized view of how the system is structured, making it easier for developers and designers to build and refine the system.
2. **Improved User Experience:**
   * Helps in planning user navigation paths, ensuring ease of use and logical flow for end-users.
3. **Development Roadmap:**
   * Acts as a guide for developers to understand the scope and features that need to be implemented.
4. **Content Planning:**
   * Helps in organizing content and identifying areas that need more detail or functionality.

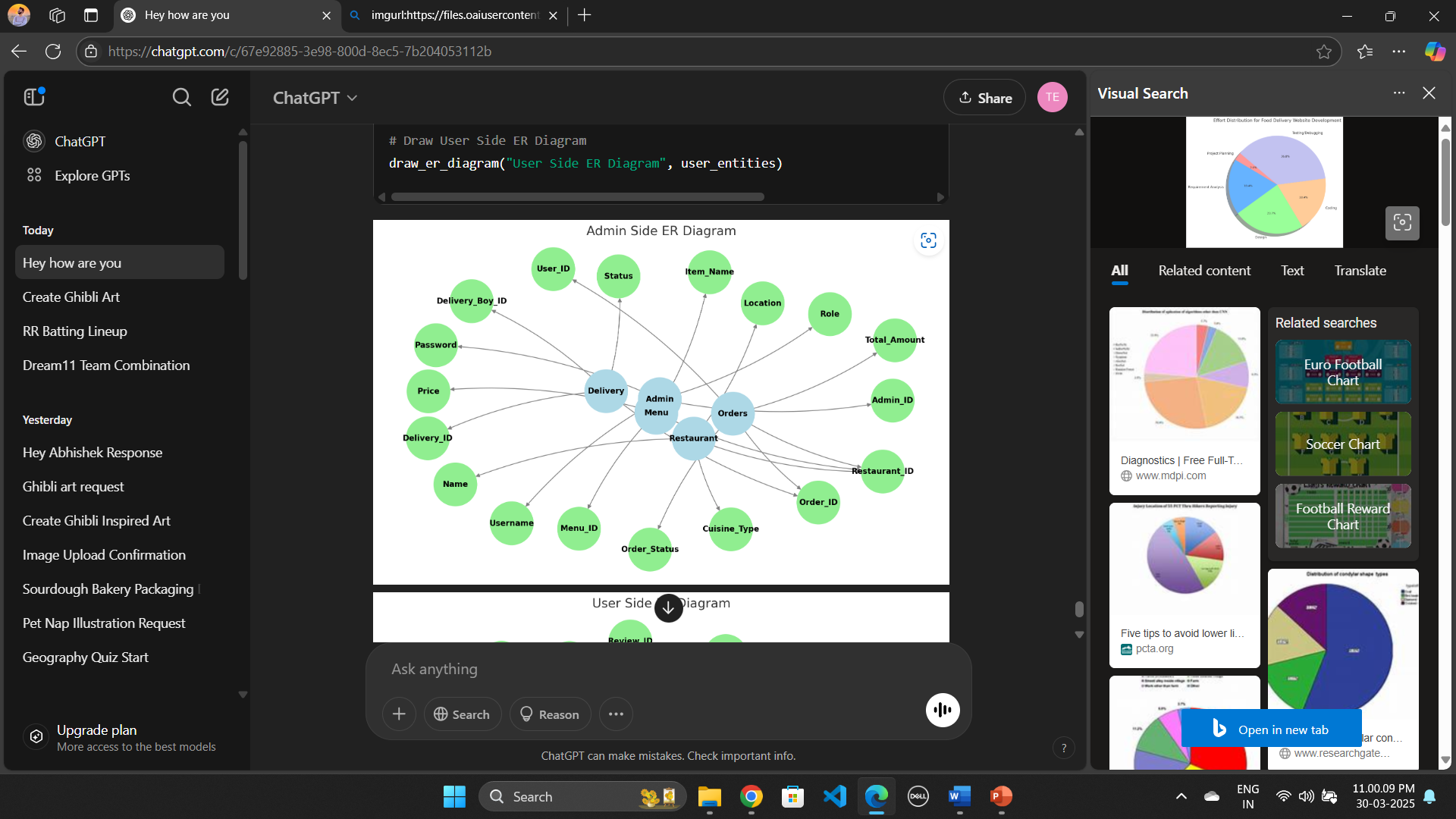
**Conclusion**

A **Navigation Map** is an essential tool for designing and understanding the structure of a system or website. It visually represents how users will navigate through the different sections, making it easier to plan, develop, and ensure an intuitive user experience.

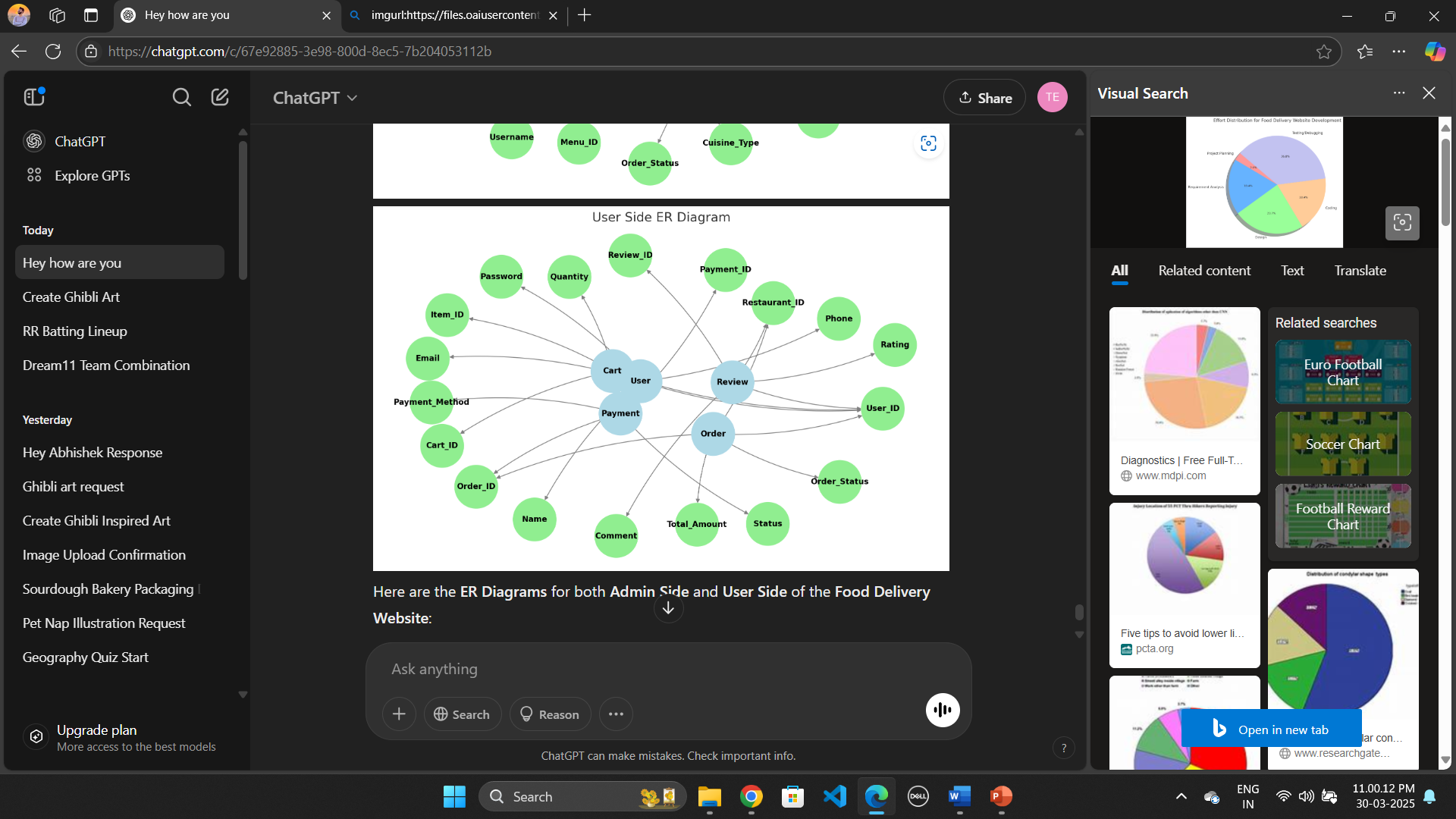
7.2 Database Diagram

A **Database Diagram** (also known as an Entity Relationship Diagram or ERD) visually represents the structure of a database. It shows the relationships between different tables (entities), as well as the attributes (fields) and key constraints of those tables. It’s essential for database design and development, providing a clear overview of how data is organized and related.

* Admin Side



* Employee Side



* **Admin Side ER Diagram** includes:
  + Admin management
  + Menu managemen
  + Order processing
  + Delivery tracking
* **User Side ER Diagram** includes:
  + User account management
  + Order placement and tracking
  + Payment handling
  + Reviews and ratings
  + Shopping cart functionality

7.3 Table Structure

Here's the **table structure** for the **Payroll Management System** database, including both **Admin** and **Employee** sides. This structure outlines the fields, data types, and constraints for each table in the system.

**Table Name: Users**  
**Use:** To store all user related data.

Note : We uses MongoDB Database

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Constraints | Description |
| Id | Int(11) | Primary Key | Unique Field |
| Name | String | Required | Specify Name. |
| Email | String | Not Null (Unique Key) | Stores EmailID. |
| Password | String | Not Null | Specify Password. |
| CartData | Object | Array | Default: empty array |

**Table Name :**

**Use :** It is used to store the information of Employee.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Constraints | Description |
| userId | String | Required | Stores the user's unique ID who placed the order. |
| items | Array | Required | Contains a list of ordered products (productId, quantity, price, etc.). |
| amount | Number | Required, Must be positive | Total amount for the order. |
| Address | Object | Array | Indicates the current status of the order. |
| Status | String | Default: "Food Processing", Enum: ["Food Processing", "Out for Delivery", "Delivered", "Cancelled"] | Default: empty array |
| Payment | Boolean | Default: false | Indicates whether the payment is completed (true/false). |
| createdAt | Date | Auto-generated by timestamps: true | Auto-generated by timestamps: true |
| updatedAt | Date | Auto-generated by timestamps: true | Auto-generated by timestamps: true |

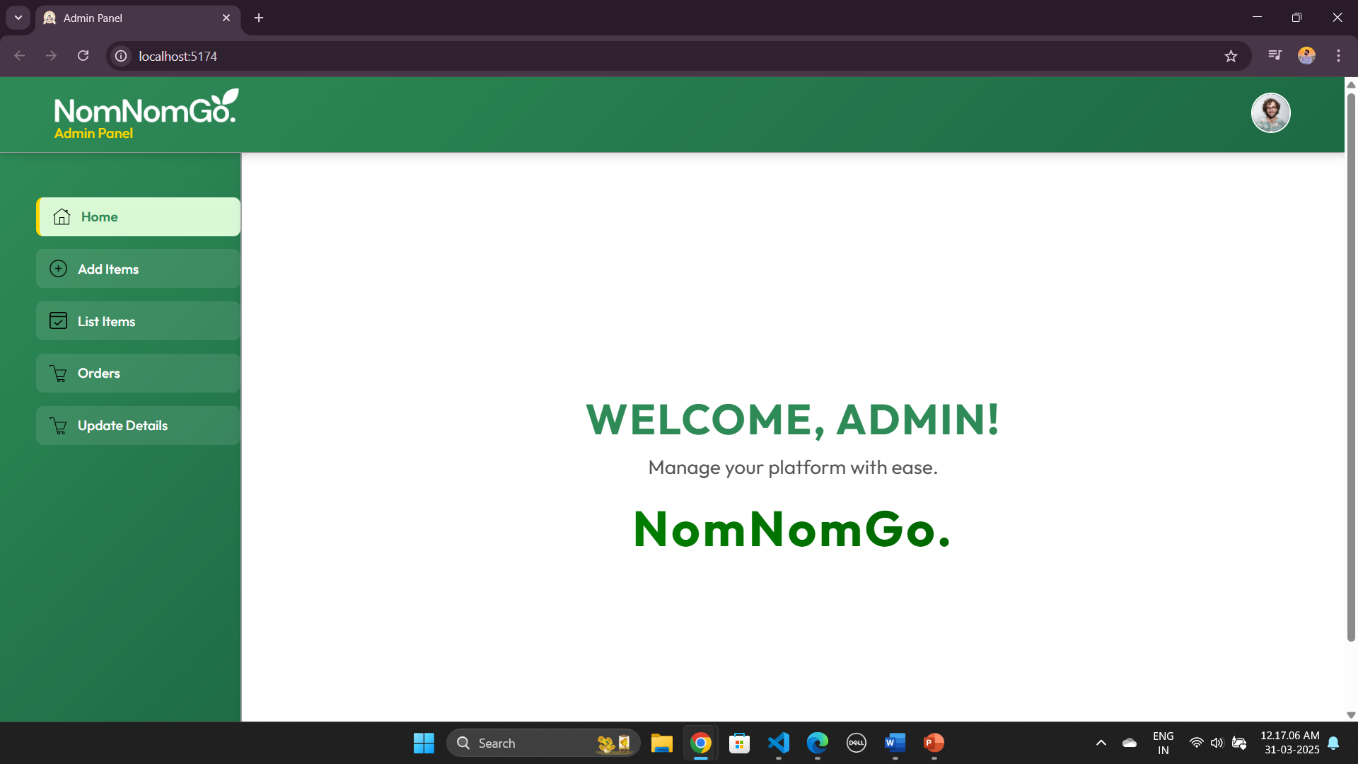
**Table Name :**

**Use :** It is used to store the information of Employee.

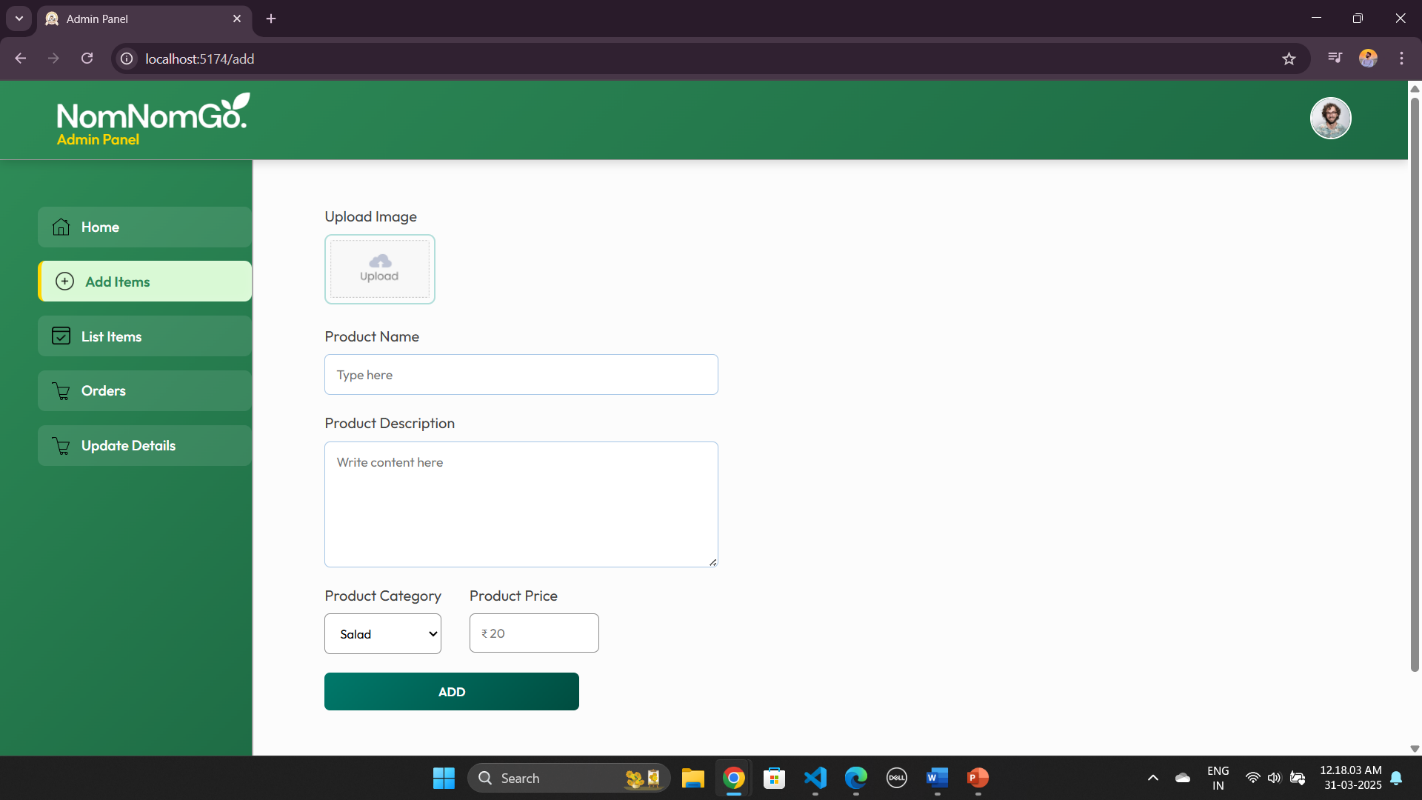
|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Constraints | Description |
| Name | String | Required | Stores the user's unique ID who placed the order. |
| description | String | Required | Provides details about the product. |
| Price | Number | Required, Must be positive | Stores the price of the product. |
| Image | String | Required | Stores the image URL or file path for the product. |
| Category | String | Required | Specifies the category to which the product belongs. |
| createdAt | Date | Auto-generated by timestamps: true | Auto-generated by timestamps: true |
| updatedAt | Date | Auto-generated by timestamps: true | Stores the timestamp when the order was last updated. |

7.4 Screen Layouts

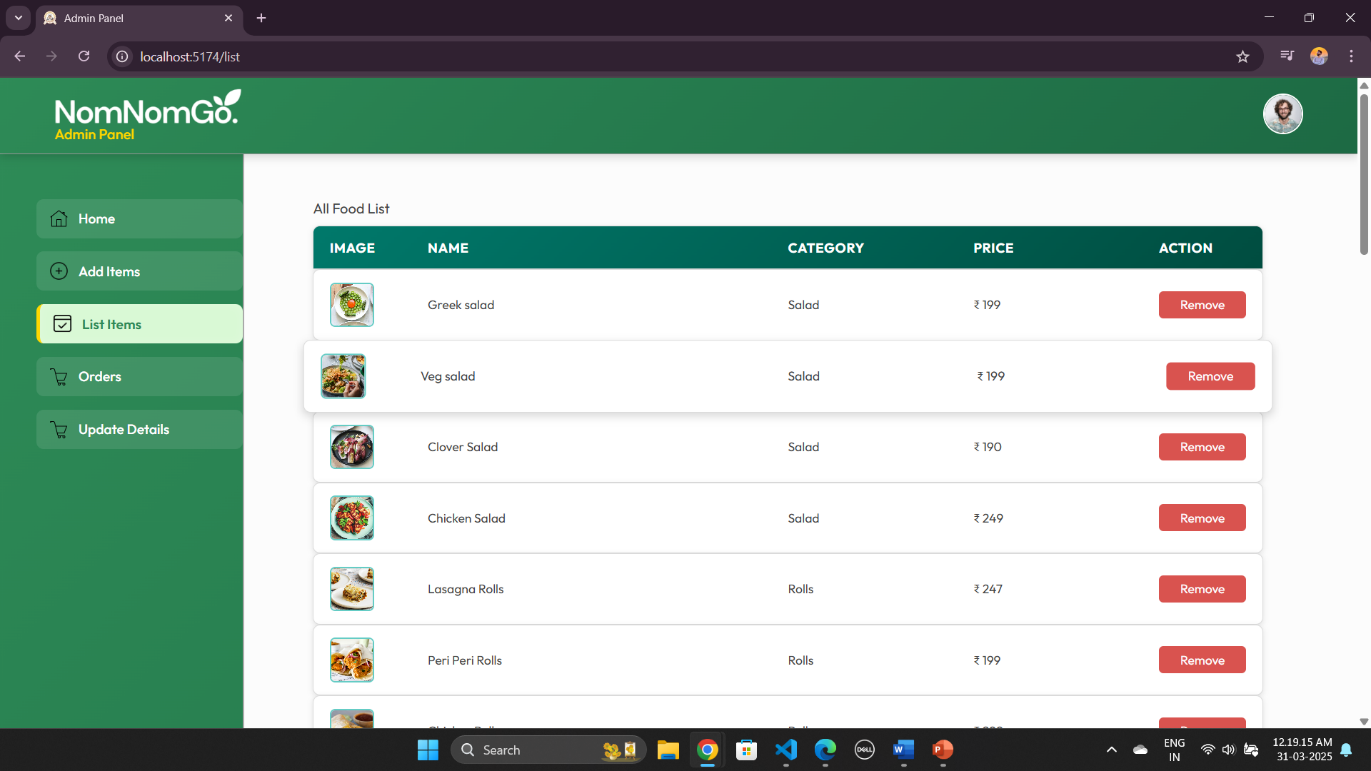
* **Admin Home Page :**



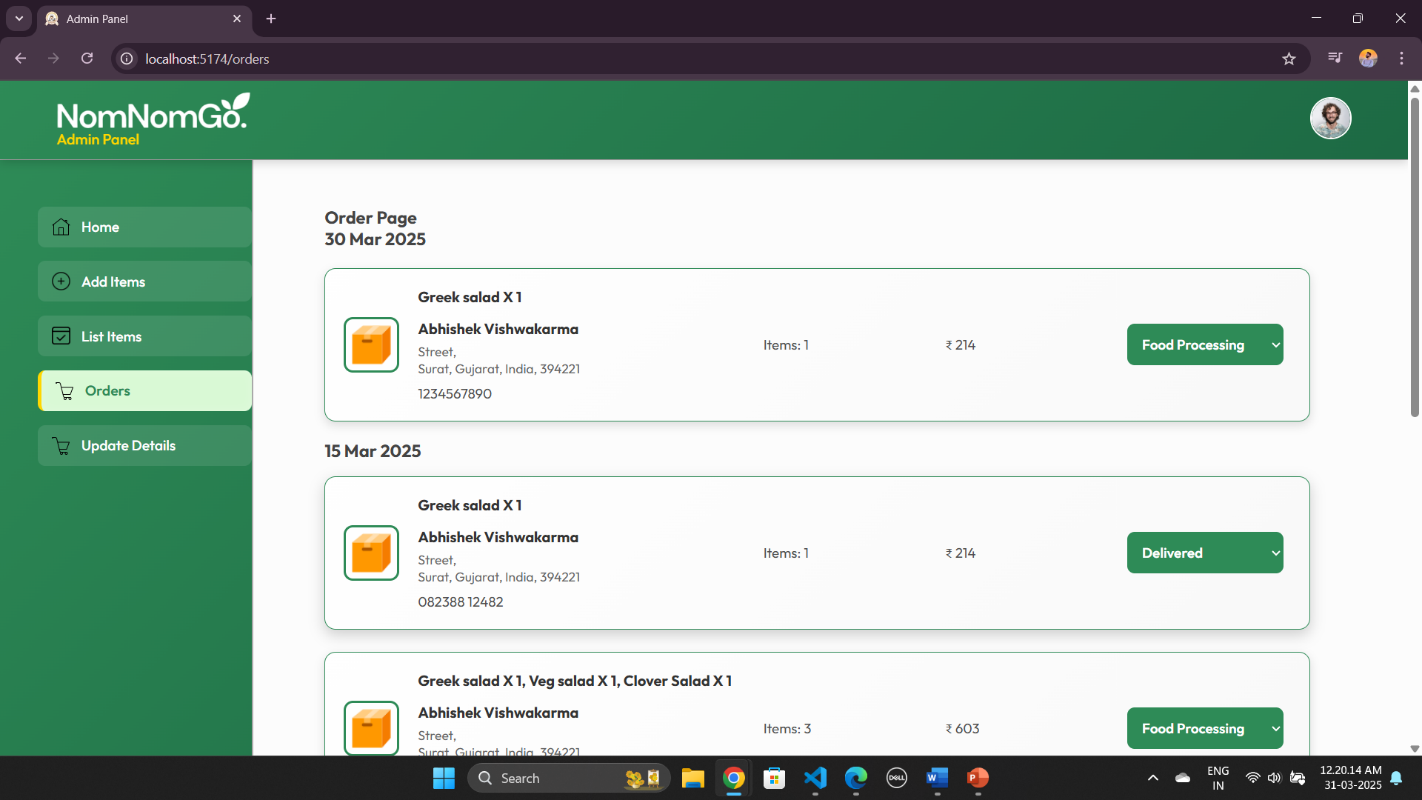
* This is home page from admin side.
* Food Add Page ( Admin Side )



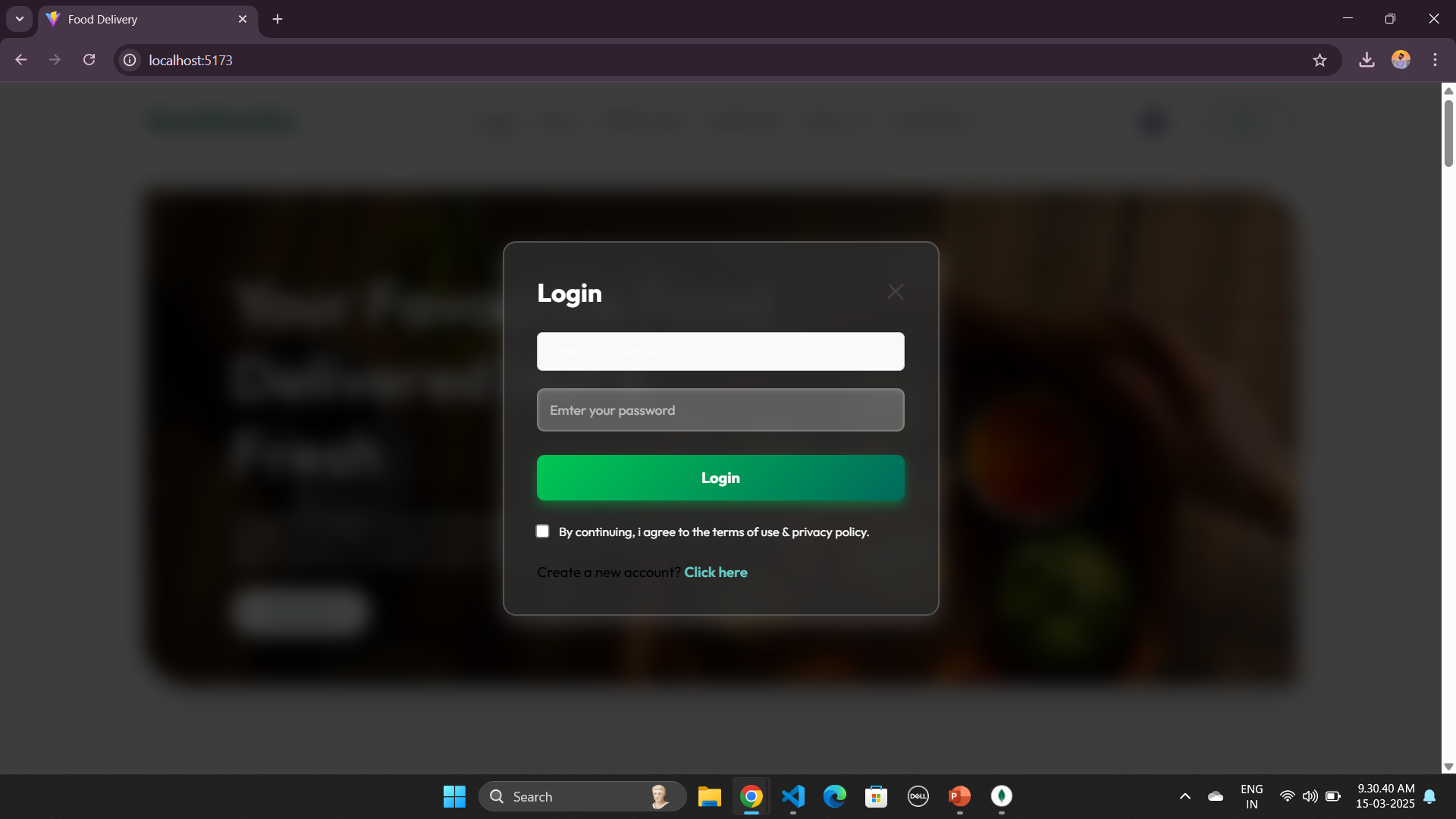
* Food List Page



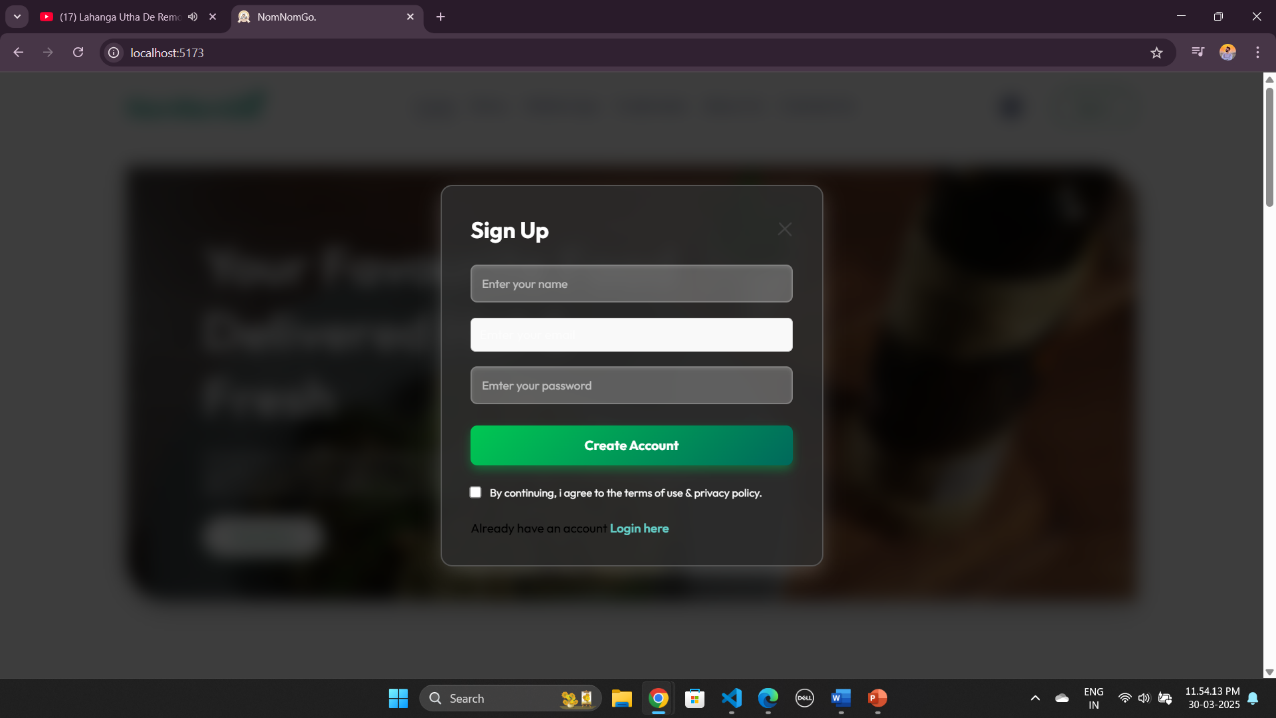
* Orders Page :



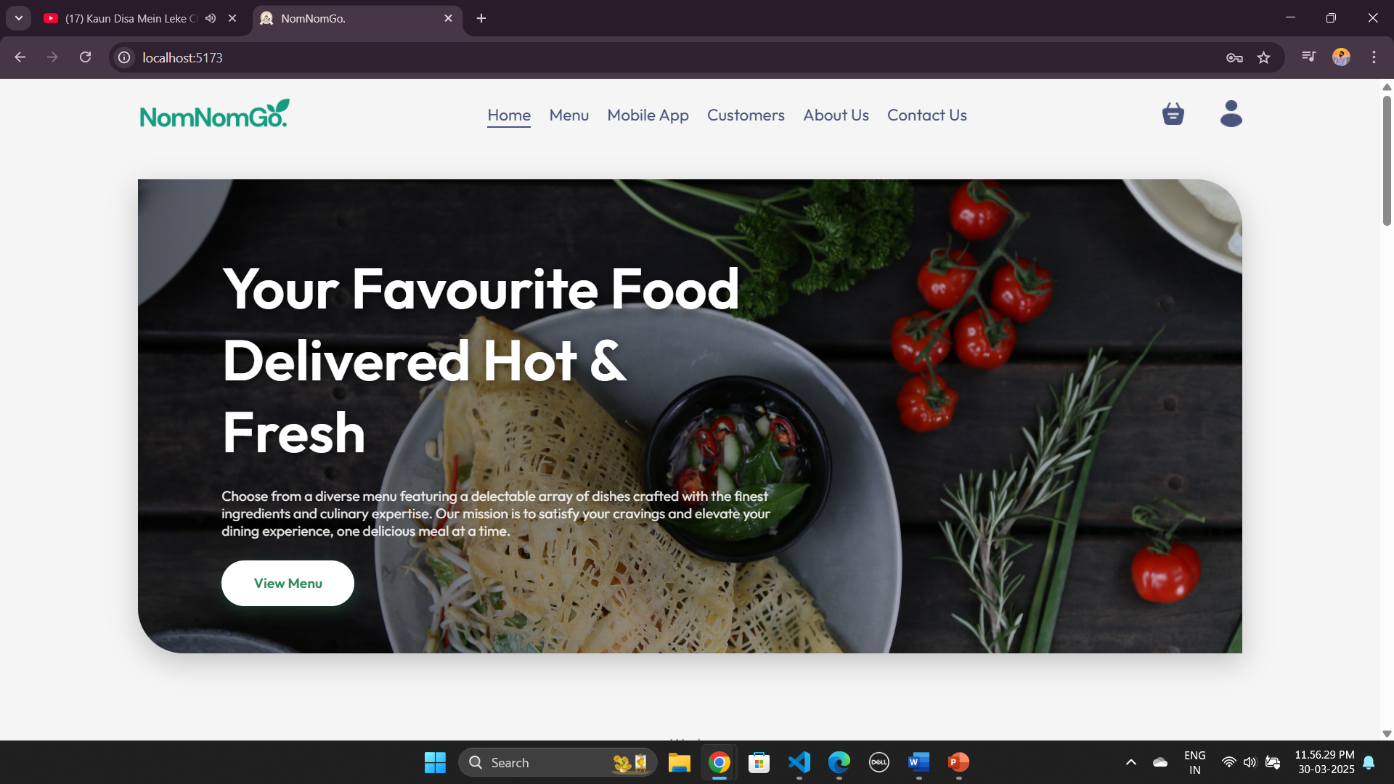
* **Login Page : Client**



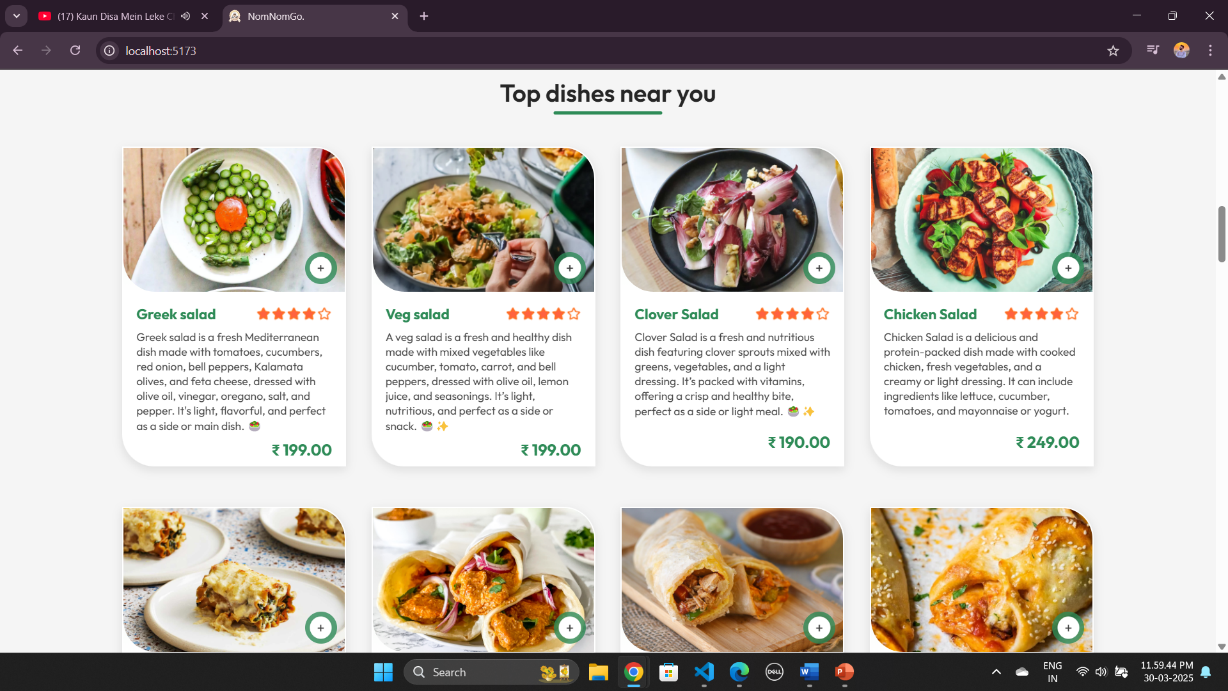
* In this page client can login in to website for authentication check.
* In any situation client forgot the password the can get password on registered mail it will display on next page.
* **Login Page : Registration Page**

****

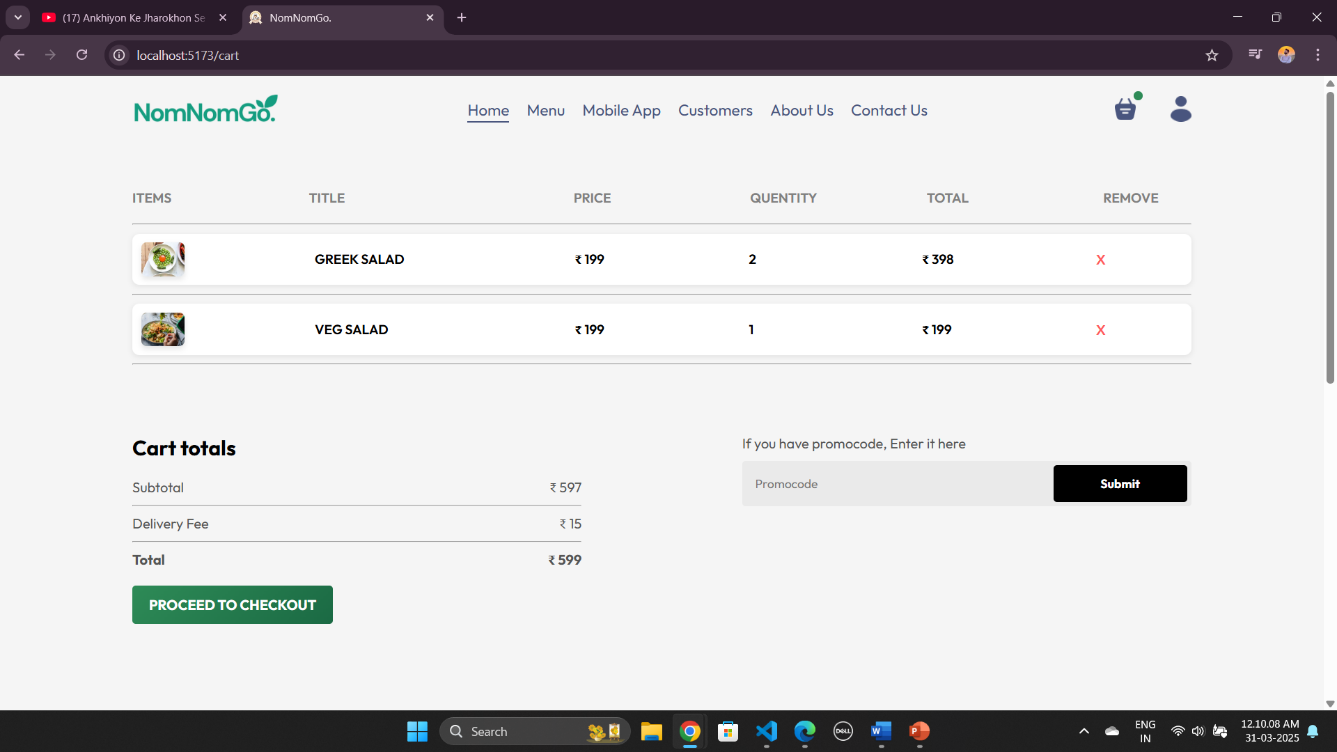
* In this page Client can register themselves.
* **Registration Page : Client Home Page**

****

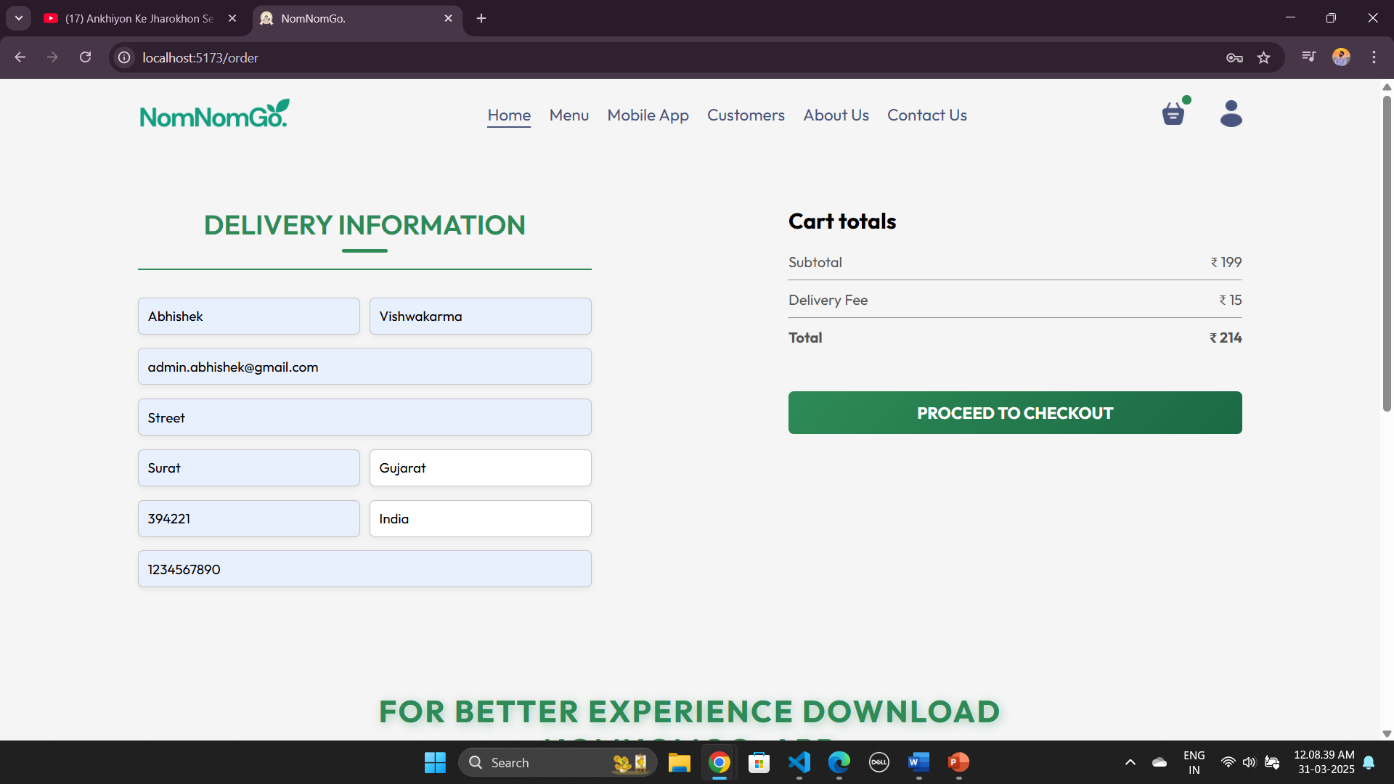
* In this page shows home page of client side.
* With navbar available options for redirecting to other pages.
* Menu Page



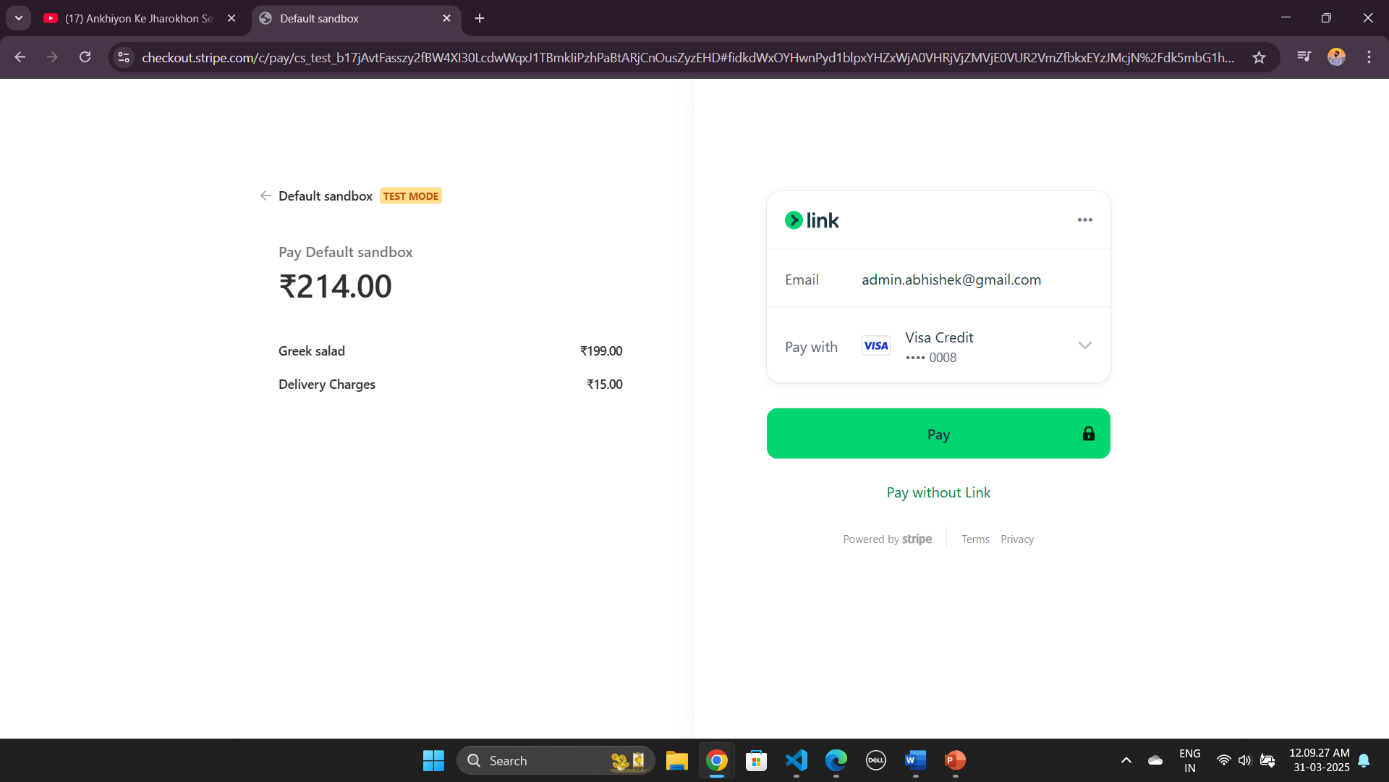
* In this page client can see different types of food available options.
* Clients Cart Page :



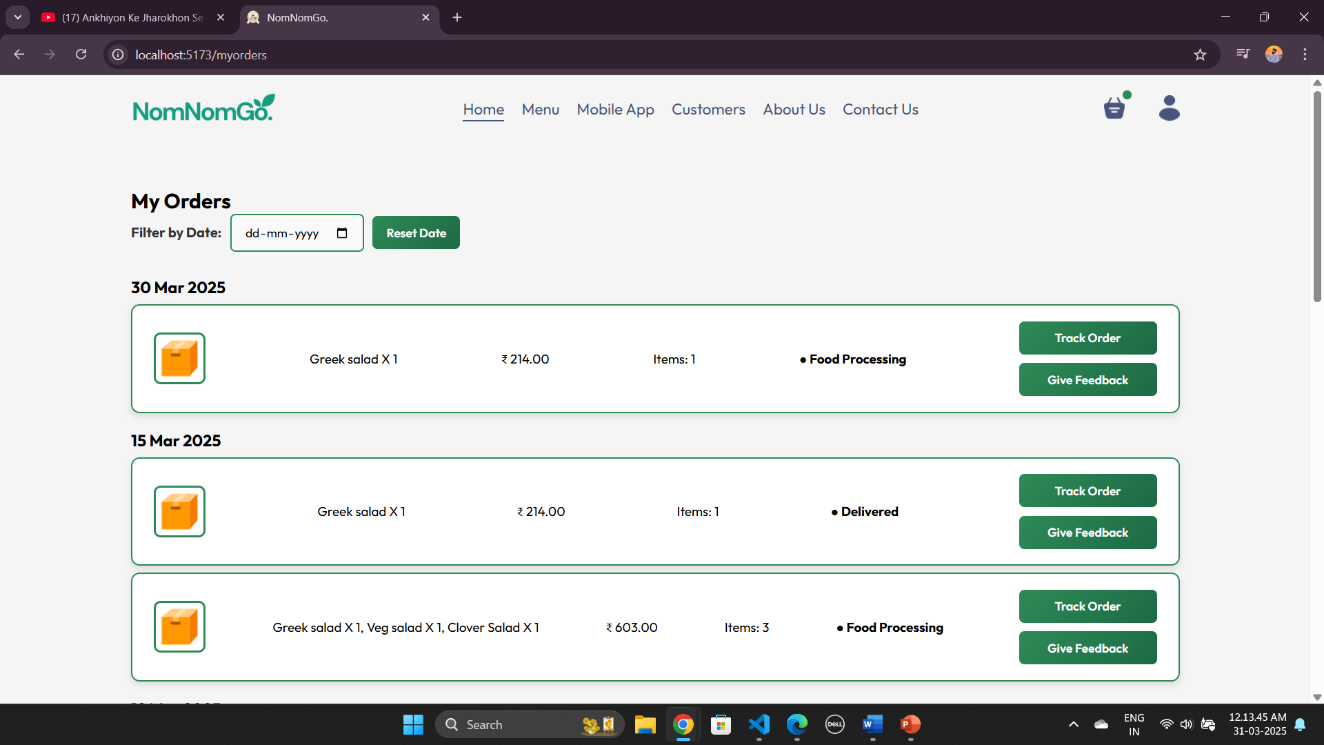
* In this page you see your selected food items with total price.
* And we provided promocode section.
* Delivery Details Page



* In this page you can have a look on delivery address details.
* Payments options.
* **Payment Gateway Page :**

****

* In this page you can see payments details.
* Total amount And payments options.
* Orders Page :



* In this page you can see your ordered food details in list view.
* With track order and give feedback options.

**8**

Chapter

**System Testing**

|  |  |
| --- | --- |
| 8.1 | Testing Introduction |
| 8.2 | Test Case |
|  |  |
|  |  |

8.1 Testing Introduction

**Software testing** is a critical phase in the software development lifecycle (SDLC) that ensures the quality, functionality, and reliability of a system before it is deployed to users. It involves executing software with the intent of identifying any bugs, defects, or gaps in requirements, and ensuring that the system meets its intended purpose.

Testing verifies that the developed software product aligns with the original requirements and specifications while ensuring that it performs efficiently under various conditions. The process not only helps in detecting defects but also ensures that the software delivers a seamless user experience.

**Importance of Testing:**

* **Error Detection: Helps identify defects in the software early in the development cycle, reducing costs for fixing bugs later.**
* **Improves Software Quality: Ensures that the product is bug-free and functions smoothly, leading to higher user satisfaction.**
* **Ensures Reliability and Performance: Testing validates that the software can handle real-world conditions, such as high user load or unexpected inputs.**
* **Compliance and Security: Testing ensures that the software meets industry standards, regulatory compliance, and is secure against vulnerabilities.**

**Conclusion**

**Testing is a vital part of software development that ensures the final product is robust, reliable, and meets user expectations. By identifying and addressing issues early, it reduces the likelihood of defects reaching the end-user, leading to a successful and well-functioning system.**

8.2 Test Case

1. **Data entry**: Test the data entry fields to make sure they are properly formatted and validated. For example, the phone no field should only accept numeric values and 10-digit number and hire date field should only accept date values.
2. **Calculation**: Test the Amount calculations to make sure they are accurate. For example, the system should correctly calculate all employee salary accurately.
3. **Security:** Test the security of the payroll system to make sure it is protected form unauthorized access. For example, the system should require users to authenticate themselves before they can access sensitive data.
4. **Reports:** Test the payroll reports to make sure they are accurate and easy to read. For example, the system should generate a report that list all employees and their pay stubs for a given period.

**9**

Chapter

**System Limitations And Future Enhancement**

Even the most advanced systems have certain limitations. In the context of a **Food Delivery Website**, there are several potential constraints and areas that could affect its overall functionality or performance.

1. **Limited Scalability**:
   * The website may struggle to handle a large number of orders simultaneously, especially during peak hours.
   * Performance issues may arise if the infrastructure is not optimized for high traffic.
2. **Customization Restrictions**:
   * The system may lack flexibility for restaurants to customize their menu options, pricing, and special offers dynamically.
   * Limited ability for users to request modifications to their food orders (e.g., removing ingredients or adding special instructions).
3. **Dependency on Internet Connectivity**:
   * Users need a stable internet connection to browse restaurants, place orders, and track deliveries.
   * Any downtime in the server or network can cause order failures and customer dissatisfaction.
4. **Limited Integration with Third-Party Services:**
   * If the system does not integrate with multiple payment gateways, users may have limited options for online transactions.
   * Lack of integration with restaurant inventory systems could lead to customers ordering out-of-stock items.
5. **Data Security Concerns**:
   * Handling sensitive customer data, such as credit card details and personal addresses, makes the system a potential target for cyber threats.
6. **Delivery Management Challenges:**
   * Inefficient tracking and assignment of delivery personnel may lead to delays and customer dissatisfaction.
   * Limited real-time tracking options for customers to monitor their orders.
7. **Limited Support for Multiple Cuisines and Languages:**
   * The system may not provide multilingual support, making it difficult for non-English speakers to navigate.

**Future Enhancements**

There are several ways a Food Delivery Website can be improved to address existing limitations and provide added value to users:

1. **Enhanced Scalability and Performance**:
   * Implement cloud-based hosting to handle high traffic during peak hours.
   * Use load balancing techniques to distribute traffic and prevent slowdowns.
2. **Advanced Customization for Users and Restaurants:**
   * Enable customers to modify their food orders (e.g., add/remove ingredients).
   * Allow restaurants to create dynamic pricing, special deals, and discounts in real time.
3. Improved Payment Gateway Integration:
   * Support for multiple payment options, including UPI, wallets, crypto payments, and BNPL (Buy Now, Pay Later) services.
   * Integration with loyalty programs and reward points systems for returning customers.
4. AI-Powered Order Recommendations:
   * Use machine learning to recommend food items based on user preferences and past orders.
   * Personalized offers and discounts based on customer behavior.
5. Real-Time Order Tracking:
   * Integration with Google Maps or GPS-based tracking for better delivery monitoring.
   * Customers should receive estimated delivery times and live tracking updates.
6. Improved Security Measures:
   * Implement end-to-end encryption for user transactions and data storage.
   * Introduce multi-factor authentication (MFA) for added security.
7. Multilingual and Multi-Currency Support:
   * Allow customers to use the platform in different languages for a better user experience.
   * Enable transactions in multiple currencies for international expansion.
8. AI-Based Chatbots and Customer Support:
   * Introduce AI-powered chatbots to assist customers with common queries and complaints.
   * Provide automated solutions for refund requests, delivery issues, and restaurant inquiries.
9. Cloud-Based Backup and Disaster Recovery:
   * Automatic data backups to prevent loss in case of system failures.
   * Quick recovery solutions to ensure continuous service availability.

**10**

Chapter

**References**

|  |  |
| --- | --- |
| 10.1 | Bibliography |
| 10.2 | Webography |

10.1 **Bibliography**

All the needed information related to my project “**Payroll Management System”** was being clumped from the following sources:

* **Books:**
* **For Frontend Development (React & JavaScript):**
* *JavaScript: The Definitive Guide*, David Flanagan, 6th Edition, 2011
* *Learning React: Modern Patterns for Developing React Apps*, Alex Banks & Eve Porcello, 2nd Edition, 2020.
* *React Up and Running*, Stoyan Stefanov, 2016
* **For Backend Development (Node.js & Express.js):**
* *Node.js Design Patterns, Mario Casciaro, 2nd Edition, 2018*
* *Web Development with Node and Express*, Ethan Brown, 2nd Edition, 2019
* **For Database (MongoDB & Mongoose):**
* *MongoDB: The Definitive Guide*, Shannon Bradshaw & Eoin Brazil, 3rd Edition, 2019
* *Mongoose for Application Development*, Simon Holmes, 2016

10.1 **Bibliography**

* **Sites** [**URL:-**](file:///D:\Food%20Delivery%20Website\Documptation%20&%20PPT\-)

**Online Resources:**

* **React.js Official Documentation:** <https://react.dev/>
* **Node.js Official Documentation:** https://nodejs.org/en/docs
* **Express.js Official Documentation:** <https://expressjs.com/>
* **MongoDB Documentation:** <https://www.mongodb.com/docs/>
* **Mongoose Documentation:** https://mongoosejs.com/docs/
* **Redux Documentation (for State Management):** https://redux.js.org/
* **Tailwind CSS Documentation (for UI Design):** <https://tailwindcss.com/docs>