RESTAURANT RESERVATION

Project report (CA3) submitted in fulfillment of the requirements for theDegree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

AKSHAT KUSHWAHA

(12214429)

To-DR.SENTHIL KUMAR J

SUBJECT-

INT222-ADVANCED WEB DEVELOPMENT

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

INDEX

1. Introduction:

The purpose of this Software Requirements Specification (SRS) document is to provide a comprehensive overview of the requirements and functionalities of an food ordering web application. The application aims to ease online shopping for customers, offering features such as adding items to a shopping cart, managing user accounts, placing orders, and tracking order statuses. The system will be developed using Node.js for the backend, with Express.js framework for server-side operations and MongoDB for database management. Additionally, Laravel Mix will be used for front-end asset compilation and optimization, while Tailwind CSS will enhance the styling of the user interface.

1.1 Purpose:

The purpose of growing this food ordering net utility is to create a sturdy and efficient platform thatenables users to have interaction in online purchasing seamlessly. By providing a user-pleasant interface and vital functionalities along with product browsing, cart management, and order processing, the software ambitions to beautify the general shopping enjoy for customers. Additionally, the machine will provide administrative equipment to control merchandise and orders effectively, catering to the desires of both clients and administrators. Through the implementation of modern net technology and best practices, the application intends to streamline the net shopping system, thereby promoting user satisfaction and easing an enterprise boom.

1.2 Scope:

The scope of the food ordering web software assignment encompasses diverse modules and functionalities designed to meet the wishes of each customer and administrator.
These consist of:

1. **User Authentication:** Implementing stable user authentication mechanisms to allow customers to sign in, log in, and manipulate their accounts.

2. **Product Management:** Providing administrators with equipment to add, edit, and cut products, in addition to controlling product classes and attributes.

3. **Shopping Cart Functionality:** Allowing clients to feature merchandise to their shopping cart, replace quantities, and as wished.

4. **Order Processing:** Easing the checkout system for clients, along with order affirmation, coping with choice, and fee integration.

5. **Order Management:** Providing directors with the potential to view, system, and update the reputation of orders, in addition to control order info and client records.

6. **User Dashboard:** Offering a customized dashboard for registered users to view order records.

7. **Administrative Dashboard:** Equipping administrators with a complete dashboard to reveal sales, control product stock, and examine patron statistics.
8. **Notification System:** Implementing a notification machine to notify users order popularity updates, account sports, and promotional offers.

9. **Optimization:** Perfecting the application for overall performance, such as quicker web page

load instances, green database queries, and responsive layout for various gadgets.

10. **Security Measures:** Implementing security features which include statistics encryption, consumer authorization, and protection against commonplace net vulnerabilities to make sure the safety of user statistics and transactions.

11. **Scalability and Maintainability:** Designing the application architecture in a scalable and maintainable manner to deal with destiny growth and updates.

By addressing those factors in the scope of the task, the food ordering net application targets to offer acomprehensive and pleasant shopping level for users while easing green management for directors.

1.3 Definition, Acronyms and Abbreviations

1. **SRS**: Software Requirements Specification - A file that outlines the practical and non-useful necessities of a software device.

2. **NPM**: Node Package Manager - A bundle manager for JavaScript programming language that helps the installation and management of software packages and dependencies.

3. **Yarn**: A package manager for JavaScript programming language that offers improvements over NPM in terms of performance, security, and reliability.

4. **CSS: Cascading Style Sheets** - A style sheet language used for describing the presentation of a document written in HTML or XML.

5. **HTML: Hypertext Markup** Language - The well-known markup language for developing internet pages and internet programs.

6. **ExpressJS**: A net application framework for NodeJS, designed for constructing net applications and APIs.

7. **EJS**: Embedded JavaScript - A templating engine that generates HTML markup with JavaScript templates.

8. **SCSS**: **Sassy CSS** - A preprocessor scripting language that is interpreted or compiled into Cascading Style Sheets.

9. **Dev Dependencies**: Dependencies required for development and testing functions, not for manufacturing use.
10. **Nodemon**: A utility that checks for modifications in NodeJS packages and mechanically restarts the server.

11. **Git**: A distributed version manages machine used for tracking modifications in supply code at some point of software program improvement.

12. **Laravel Mix**: An asset compilation tool constructed on pinnacle of webpack, used for compiling JavaScript, CSS, and different assets.

13. **Tailwind CSS**: A application-first CSS framework for building custom designs without having to leave your HTML.

14. **MongoDB**: A NoSQL database software that uses JSON-like files with schema.

15. **Axios**: A promise-based totally HTTP patron for the browser and NodeJS.

16. **PassportJS**: An authentication middleware for NodeJS that helps many authentication techniques, along with username and password, OAuth, and extra.

17. **CRUD**: Create, Read, Update, Cut - Basic operations for chronic storage.

18. **HTTP**: Hypertext Transfer Protocol - The protocol used for transmitting information over the World Wide Web.

19. **API**: Application Programming Interface - A set of policies and protocols that allows one-of-a-kind software program applications to talk with every other.

20. **URL**: Uniform Resource Locator - A connection with a web useful resource that specifies its place on a computer network and the mechanism for retrieving it.

21. **JavaScript**: A programming language this is normally used for growing interactive results within web browsers.

2. General Description

This section of the SRS provides an overview of the general factors that influence the product and its requirements. It aims to contextualize the project by considering various aspects that affect its development and functionality. It is important to note that this section does not list specific requirements but rather provides background information to aid in understanding the later detailed requirements.

2.1 Product perspective

This subsection of the SRS gives context for the product by considering its relationship with other related products or projects. It is ambitioned to explain how the proposed food ordering internet software fits inside the broader panorama of comparable systems and projects.

Comparison with Existing Solutions:

The food ordering web application may be evolved as a standalone platform dedicated to easing onlinepurchasing for users. While there are various present food ordering solutions to be had in the market, every with its own set of features and functionalities, the proposed application pursuits to distinguish itself by providing a completely unique aggregate of user-pleasant interface, sturdy overall performance, and complete feature set.

Integration with External Services:

In addition to standalone operation, the food ordering internet utility may additionally combine withoutside services and systems to decorate its capability and user enjoy.

Compatibility with Industry Standards:

The food ordering web utility will adhere to enterprise standards and satisfactory practices to make sure compatibility and interoperability with existing structures and technologies. This consists of compliance with web standards which include HTML5, CSS3, and JavaScript, as well as adherence to protocols and specs for information alternate, safety, and performance optimization.

Scalability and Extensibility:

As the food ordering market continues to evolve and grow, the proposed application needs to be designed toscale and adapt to changing requirements and consumer demands. This includes enforcing scalable architecture and modular design ideas that enable the addition of recent capabilities and functionalities within the destiny.

Differentiation and Competitive Advantage:

While the food ordering internet utility may percentage similarities with current solutions, it will striveto differentiate itself through innovation, consumer enjoy, and cost proposition. By figuring out and addressing gaps in the marketplace, in addition to using rising technology and tendencies, the application aims to carve out a gap and set up a competitive gain in the crowded food ordering landscape.

2.2 Product Functions

This subsection of the SRS outlines the primary capabilities that the food ordering net application will conduct. These capabilities embody the core capabilities and skills of the software, imparting acomprehensive overview of its meant conduct and capability.

User Authentication and Authorization:

Allow customers to check in for an account.
Enable customers to log in securely with their credentials.
Implement role-primarily based access manipulate to limit positive functionalities to legal customers.

Product Management:

Enable directors to add, edit, and cut products.
Support categorization and business enterprise of merchandise into distinctive classes and subcategories.
Provide functionalities for managing product attributes, which include size, coloration, and quantity.

Product Browsing:

Allow customers to browse through available products.
Facilitate navigation through product classes.

Shopping Cart Management:

Enable users to add products to their shopping cart.
Support the amendment of portions and elimination of gadgets from the cart.
Display a precis of gadgets in the cart and the full order quantity.

Order Processing:

Facilitate the checkout system for customers, which includes imparting transport and billing facts.
Validate consumer inputs and ensure the accuracy of order details.
Generate order confirmation and offer users with order monitoring data.

Order Management:

Provide directors with gear to view and control incoming orders.
Enable order processing functionalities, consisting of updating order popularity and staining orders.

Support order success and integration with shipping companies for monitoring functions.

User Account Management:

Enable users to view order history and music the status in their orders.

2.3 User Characteristics

This subsection of the SRS describes the overall traits of the eventual customers of the product that will have an impact on the necessities. Understanding the consumer demographics, possibilities, and behavior eases in tailoring the software program to meet their needs correctly.

Customers:

Preferences: Users might also have preferences concerning product categories, brands, and pricing.
Behavior: Customers might also highlight one-of-a-kind buying behaviors, including surfing, looking, and impulse buying.
Technical Proficiency: Users might also have varying levels of technical ability, ranging from beginner to experienced internet users.
Expectations: Customers expect a user-friendly interface, easy navigation, secure transactions, and prompt customer service.

Technical Proficiency: Administrators are expected to have a better level of technical ability in comparison to everyday customers, as they need to interact with backend systems and conduct administrative tasks.
Efficiency: Administrators require green gear and functionalities to streamline their responsibilities and control the platform correctly.
Access Levels: Depending on their roles, administrators may need exceptional get admission to range and permissions to perform precise moves, consisting of including products, processing orders, and generating reviews.

Guest Users:

Limited Access: Guest customers are site visitors who have now not registered for an account on the platform.
Browsing and Shopping: Guest users can browse merchandise, add objects to their cart, and go ahead to checkout without creating an account.
Conversion Rate Optimization: The platform must provide a continuing guest checkout experience to inspire conversion and minimize boundaries to buy.

Mobile Users:

On-the-Go Access: Mobile users get admission to the food ordering platform the usage of smartphones orpills.
Responsive Design: The platform needs to be perfected for cellular devices, with responsive design and intuitive navigation.
Mobile-Friendly Features: Mobile customers may pick capabilities inclusive of contact gestures, simplified forms, and brief access to crucial functionalities.
Understanding these person characteristics allows the improvement crew to prioritize capabilities, design intuitive consumer interfaces, and tailor the software program to meet the precise needs and expectancies of different consumer corporations. By catering to the various needs of customers, administrators, visitor customers, and mobile customers, the food ordering platform can decorate consumerpleasure and drive engagement and conversions.

2.4 General Constraints

This subsection of the SRS outlines the overall constraints that may limit the developer's options for designing the gadget. These constraints encompass several factors such as technical barriers, regulatory requirements, budgetary constraints, and organizational rules, which affect the improvement process and shape the final product.

Technological Constraints:

Compatibility: The machine ought to be like minded with technologies or structures, which includes running systems, net browsers, or cellular gadgets.
Integration Requirements: Integration with present structures or third-party offerings can also impose constraints on generation selections and implementation methods.
Performance Requirements: Performance constraints, which include reaction times, throughput, and useful resource use, must be considered all through device layout and implementation.

Data Protection Regulations: Compliance with facts safety legal guidelines and regulations, including GDPR (General Data Protection Regulation) or HIPAA (Health Insurance Portability and Accountability Act), might also impose constraints on statistics coping with, garage, and safety features.

Industry Standards: Adherence to enterprise requirements and fine practices, which include PCI DSS (Payment Card Industry Data Security Standard) for fee processing, may dictate specific necessities for system design and implementation.

Budgetary and Resource Constraints:

Financial Limitations: Budget constraints may also restrict the assets available for system improvement, along with employees, generation, and infrastructure.

Time Constraints: Project cut-off dates and time-to-marketplace issues may additionally impose constraints at the improvement timeline, requiring efficient resource allocation and prioritization of tasks.

Organizational Constraints:

Internal Policies: Organizational regulations and recommendations, inclusive of IT governance frameworks or safety protocols, may additionally impose constraints on machine architecture, technology selection, and implementation practices.

Stakeholder Requirements: Requirements and choices of stakeholders, which include clients, management, and give up-customers, can also affect design decisions, and constrain device development.

Environmental Constraints:

Infrastructure Limitations: Constraints related to the bodily or environmental conditions in which the device will run, including community bandwidth, server capability, or geographical area, may additionally impact machine layout and overall performance.

Usability and Accessibility Constraints:

Usability Requirements: Usability constraints, consisting of accessibility requirements and suggestions (e.G., WCAG), may additionally dictate specific design considerations to make sure the machine is usable by people with disabilities.

Multilingual Support: Requirements for multilingual guide may additionally impose constraints on consumer interface design, content material localization, and internationalization efforts.

Security Constraints:

Security Policies: Compliance with organizational safety guidelines and requirements, in addition to enterprise-precise protection necessities, may additionally impose constraints on gadget architecture, facts encryption, and get entry to manipulate mechanisms.

By figuring out and addressing those general constraints prematurely, the improvement group can correctly manipulate mission dangers, make informed layout decisions, and make certain a success delivery of the gadget within the designated constraints and requirements.

3.Technologies Used

Front-End Technologies-

HTML:

HTML (Hypertext Markup Language) is used to create the structure and content of web pages. The code contains HTML templates and forms to handle the user interface and interactions, such as login forms, displaying data, and a smooth scroll to sections.

CSS:

The website uses CSS for styling elements such as buttons, form elements, and text. Custom styles are added using classes to manage appearance and layout. Classes like bg-white, text-lg, and rounded suggest the use of utility classes for styling.

JavaScript:

JavaScript is heavily used for dynamic functionality such as handling events, making HTTP requests, and managing DOM manipulation.

The script contains event listeners to handle user interactions (e.g., button clicks), update the user interface (e.g., adding items to the cart), and make AJAX requests (e.g., for order status updates).

Noty:

Noty is a library for creating notification pop-ups. It is used in the code to show messages to the user, such as when an item is added to the cart or when an error occurs.

Moment.js:

Moment.js is a library used for date and time manipulation. It is used in the code to format date and time data (e.g., displaying the order creation time).

Socket.io:

Socket.io is a library that allows real-time communication between the server and the client. It is used in the code for handling real-time events, such as updating order statuses and receiving new orders.

Axios:

Axios is a JavaScript library for making HTTP requests. The code uses Axios to fetch data from the server, such as fetching orders and making requests to update the cart.

Back-End Technologies-

Express:

Express is a web application framework for Node.js. The code suggests that the website uses Express for handling server-side routing and managing HTTP

requests and responses.

MongoDB and Mongoose:

MongoDB is a NoSQL database used for storing data such as user information and orders. Mongoose is an ORM (Object-Relational Mapping) library for MongoDB, which is used in the code to interact with the database.

Passport.js:

Passport.js is a library for authentication. The code suggests that the website uses Passport.js for handling user authentication and sessions.

EJS:

EJS (Embedded JavaScript) is a templating engine used for generating HTML pages dynamically on the server side based on data and logic.

Express-Session:

Express-Session is a session management middleware for Express, used in the website to handle user sessions.

Connect-Mongo:

Connect-Mongo is used to store session data in a MongoDB database, which helps with session persistence and scalability.

dotenv:

dotenv is a Node.js package for loading environment variables from a .env file, allowing the application to keep sensitive information, such as database connection strings and secret keys, secure.

Development Tools

Laravel-Mix:

Laravel-Mix is a wrapper around Webpack that simplifies configuration and builds tasks for the application.

Webpack:

Webpack is a module bundler used for bundling and processing JavaScript, CSS, and other assets.

Nodemon:

Nodemon is a utility that automatically restarts the Node.js server when file changes are detected, aiding in development efficiency.

4.Key Features Of The App

3.1 External Interface Requirements

3.1.1 User Interfaces

This section describes the user interfaces (UI) of the food ordering web application, including the layout, design elements, and interactions available to users. The UI design aims to provide an intuitive and visually appealing experience for users, easing seamless navigation and interaction with the platform.

1. Homepage

1.1 Header

The header section has the application logo, navigation menu, and user authentication options (login/register).
Upon login, the user's account options are displayed.

1.2 Main Content

The main content section highlights featured products, promotions, and new arrivals.
Products are displayed in a grid layout with product images, names, prices, and "Add to Cart" buttons.

1.3-Footer

The footer section includes links to important pages such as About Us, Contact, Terms of Service, and Privacy Policy.
Social media icons are provided for users to follow the application on various platforms.

2. Product Listings Page

2.1 Product Cards

Each product is displayed as a card with an image, name, price, and "Add to Cart" button.
Hovering over a product card reveals more information such as product ratings and reviews.

3. Product Details Page

3.1 Product Image and Description

The product details page displays a larger image of the selected product along with a detailed description.
More information such as product specifications, sizes, and colors are provided.

3.2 Quantity Selector and "Add to Cart" Button

Users can select the desired quantity of the product using a dropdown selector.
An "Add to Cart" button allows users to add the selected product to their shopping cart.

4. Shopping Cart

4.1 Checkout Button
A prominent "Checkout" button allows users to go ahead to the checkout process to complete their purchase.

5. Checkout Process
5.1 Shipping and Billing Information
Users are prompted to enter shipping and billing information, including name, address, email, and payment details.

5.2 Order Summary
A summary of the order, including the list of items, subtotal, taxes, shipping fees, and total amount, is displayed for review.

5.3 Confirmation Page
Upon successful completion of the checkout process, users are directed to a confirmation page with details of their order and a confirmation message.

6. Account Management

6.1 Order History
A dedicated section allows users to view their order history, including past purchases, order status, and tracking information.
These user interface descriptions provide a comprehensive overview of the layout, features, and interactions available to users across different pages of the food ordering web application. The UIdesign aims to enhance user experience and ease seamless navigation and interaction with the platform.

3.1.2 Hardware Interfaces
This section outlines the hardware components with which the food ordering web application interacts. While the application primarily runs in a web-based environment, there may be interactions with specific hardware devices or interfaces for certain functionalities.

1. Web Browser
The food ordering web application is primarily accessed through web browsers on various devices,including desktop computers, laptops, tablets, and smartphones. The application's user interface isdesigned to be compatible with modern web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

2. Mobile Devices
Users may access the food ordering web application using mobile devices, including smartphones and tablets. The application's responsive design ensures best display and usability across different screensizes and resolutions.

3. Point-of-Sale (POS) Systems
For physical retail locations or omnichannel operations, the food ordering platform may integrate withpoint-of-sale (POS) systems for inventory management, order processing, and payment processing. Integration with POS systems enables seamless synchronization of online and offline sales channels.

5. Payment Terminals

The food ordering platform meets with payment terminals or payment gateways for processing onlinepayments securely. Integration with payment terminals allows users to make payments using credit cards, debit cards, and other electronic payment methods.

6. Shipping Label Printers

For order fulfillment and shipping operations, the food ordering platform may integrate with shipping label printers for printing shipping labels and order invoices. Integration with shipping label printersstreamlines the order fulfillment process and ensures correct shipping information.

7. Customer Support Tools

Customer support agents may use hardware devices such as computers, telephones, and headsets to interact with customers and aid with order inquiries, product recommendations, and issue resolution. Integration with customer support tools enables seamless communication and collaboration between customer support teams and users.

These hardware interfaces ease the operation and functionality of the food ordering web applicationacross various devices and environments, ensuring seamless integration with existing hardware systems and devices used in retail, fulfillment, and customer support operations.

3.1.3 Software Interfaces

This section describes the software components and systems with which the food ordering web application interacts. These interfaces enable communication, data exchange, and integration with external software systems, services, and platforms to enhance the functionality and capabilities of the application.

1. Payment Gateways

The food ordering web application interfaces with payment gateways to ease secure online transactions. Commonly used payment gateways include PayPal, Stripe, Square, and Authorize.Net. Integration with payment gateways enables users to make payments using credit cards, debit cards, and other electronic payment methods securely.

3. Content Management Systems (CMS)

The food ordering web application may meet with content management systems such as WordPress,Drupal, or Joomla for managing website content, blog posts, and marketing materials. Integration with CMS platforms enables seamless content publishing and management within the food orderingplatform.

4. Customer Relationship Management (CRM) Systems

Integration with customer relationship management (CRM) allows the food ordering platform to manage customer interactions, track customer inquiries, and analyze customer behavior. Integration with CRM systems enables personalized marketing campaigns, customer segmentation, and lead management.

5. Inventory Management Systems

Integration with inventory management systems enables real-time inventory tracking, stock level updates, and order synchronization across multiple sales channels.

6. Analytics and Reporting Tools

The food ordering web application may meet with analytics and reporting tools such as Google Analytics, Adobe Analytics, or Mix panel for tracking website traffic, user behavior, and sales performance. Integration with analytics tools enables data-driven decision-making, performance monitoring, and optimization of marketing strategies.

8. Social Media Platforms

For social media marketing and engagement, the food ordering web application interfaces with socialmedia platforms such as Facebook, Instagram, Twitter, and Pinterest. Integration with social media platforms enables sharing of products, user-generated content, and promotional offers, as well as tracking of social media engagement and referrals.

These software interfaces ease the integration of the food ordering web application with external software systems and platforms, enabling enhanced functionality, data exchange, and automation of business processes. Integration with third-party software systems enhances the user experience, streamlines operations, and drives business growth.

3.1.4 Communications Interfaces

This section describes the communication protocols and interfaces used by the food ordering web application to interact with external systems, services, and users. These interfaces ease data exchange, message transmission, and interaction between different components of the application and external entities.

1. HTTP/HTTPS Protocol

The food ordering web application communicates with clients (web browsers) using the Hypertext Transfer Protocol (HTTP) or its secure variant, HTTPS. HTTP/HTTPS protocols enable the transmission of web pages, images, scripts, and other resources between the server and client devices over the internet.

2. RESTful API

For integration with external systems and services, the food ordering platform exposes a Representational State Transfer (RESTful) API. RESTful APIs allow external applications to access and manipulate resources within the food ordering platform, such as products, orders, and user accounts, using standard HTTP methods (GET, POST, PUT, DELETE).

3. WebSocket Protocol

The food ordering web application uses the WebSocket protocol for real-time communication and messaging between the server and client devices. WebSocket allows bidirectional communication channels to be set up between the server and client, enabling instant updates, notifications, and live chat functionality within the application.

7. Social Media APIs

Integration with social media APIs such as Facebook Graph API, Twitter API, and Instagram API enables the food ordering platform to interact with social media platforms programmatically. Social

media APIs allow for sharing of product listings, user-generated content, and promotional offers on social media channels directly from the food ordering platform.

8. Webhooks

Webhooks offer a mechanism for the food ordering platform to receive real-time notifications andupdates from external systems and services. Webhooks enable event-driven communication, allowing the platform to react to events such as new orders, inventory updates, or payment confirmations from external sources.

These communications interfaces enable the food ordering web application to interact with external systems, services, and users effectively, easing data exchange, real-time communication, and automation of business processes. By using these interfaces, the platform enhances user experience, improves operational efficiency, and enables seamless integration with third-party services and platforms.

3.2 Functional Requirements

Functional requirements specify the behavior and functionality of the food ordering web application.This section outlines the inputs, processing logic, outputs, and error handling mechanisms for each functional requirement.

Introduction

The functional requirements describe the specific actions and behaviors that the food ordering webapplication must perform to meet user needs and business aims.

Inputs

Inputs are the data or information provided to the system to start or trigger specific actions or processes. These inputs are essential for executing various functionalities of the food ordering application.

User Registration Information:

Inputs include user-provided information such as name, email address and password during the registration process.
Added inputs may include user preferences, profile settings, and communication preferences.
Product Details:

Inputs include product information such as name, description, price, quantity, and images entered by administrators during product management.
Product attributes, categories, and tags may also serve as inputs for organizing and categorizing products.

Order Details:

Inputs consist of order-related information such as selected products, quantities, shipping address, billing details, and payment method entered by users during the checkout process.
Processing
Processing refers to the actions, computations, or transformations performed by the system in response to inputs received. It involves executing business logic, performing calculations, and updating data based on the provided inputs.

User Authentication and Authorization:

Processing involves validating user credentials during the login process and verifying user identity for accessing protected resources.
Authorization logic decides the permissions and access levels granted to authenticated users based on their roles and privileges.
Product Management:

Processing includes adding, editing, and cutting products from the database based on user input.
Data validation ensures that product details are correct and consistent before being saved to the database.

Order Processing:

Processing involves calculating the total order amount, including taxes and shipping fees, based on the selected products and shipping destination.
Inventory management logic updates product quantities and availability status after an order is placed to prevent overselling.
Outputs
Outputs are the results, responses, or outcomes produced by the system because of processing inputs. These outputs provide users with information, feedback, or confirmation about their actions or requests.

User Registration Confirmation:

Output includes a confirmation message or email sent to users upon successful registration, welcoming them to the platform.
User registration confirmation may also include instructions for account activation or verification if needed.

Product Listings and Details:

Outputs consist of product listings, descriptions, images, and prices displayed to users for browsing and selection.
Product details pages provide more information such as product specifications, reviews, and related products.

Order Confirmation and Receipt:

Output includes an order confirmation page displayed to users after successful checkout, summarizing the order details and providing a receipt.
Users receive order confirmation emails having order details, payment receipts, and estimated delivery dates.

Error Handling

Error handling mechanisms address situations where the system encounters exceptions, invalid inputs, or failures during processing. These mechanisms ensure graceful error recovery, user notification, and system stability.

Invalid User Credentials:

Error handling notifies users of invalid login credentials and prompts them to retry or reset their password if necessary.
User authentication failures are logged for security auditing and monitoring purposes.
Out-of-Stock Products:

Error handling notifies users when selected products are out of stock and prompts them to remove or update their selections.
Users may be provided with alternative product suggestions or notified when the selected products become available again.

Payment Processing Errors:

Error handling addresses payment processing errors such as declined transactions, expired payment methods, or insufficient funds.
Users are prompted to review and correct payment information or select an alternative payment method to complete the transaction.
These functional requirements define the inputs, processing logic, outputs, and error handling mechanisms necessary for the food ordering web application to fulfill its intended functionalitieseffectively and provide users with a seamless shopping experience.

3.3 Non-Functional Requirements

Non-functional requirements specify the quality attributes, characteristics, and constraints that govern the behavior of the food ordering web application. This section outlines the non-functionalrequirements related to performance, reliability, availability, security, maintainability, and portability.

Performance
Response Time:

The food ordering web application shall respond to user interactions, such as page loads and formsubmissions, within 3 seconds on average.

Scalability:

The application architecture shall support horizontal scaling to accommodate increasing user traffic and workload demands.
Database and server resources shall scale dynamically to keep consistent performance during peak usage periods.
Reliability

Fault Tolerance:

The food ordering platform shall be resilient to system failures, hardware faults, and networkdisruptions.
Fault-tolerant mechanisms such as redundant servers, data backups, and failover strategies shall be implemented to minimize service downtime.

Error Handling:

The application shall manage errors gracefully, providing informative error messages and guiding users towards corrective actions.
Error logging and monitoring mechanisms shall capture and report system errors for troubleshooting and analysis.

Availability

Uptime Requirement:

The food ordering web application shall strive to achieve a minimum uptime of 99.9% over a 30-dayperiod, excluding scheduled maintenance windows.
Planned maintenance activities shall be communicated to users in advance through notifications and announcements.

Redundancy:

Redundant server instances, load balancers, and database replicas shall be deployed to ensure high availability and minimize service disruptions.
Failover mechanisms shall automatically redirect traffic to healthy server instances if hardware or software failures.

Security

Data Encryption:

User authentication credentials, payment information, and sensitive personal data shall be encrypted using industry-standard encryption algorithms during transmission and storage.
Secure Socket Layer (SSL) or Transport Layer Security (TLS) protocols shall be enforced to encrypt data exchanged between the client and server.

Access Control:

Role-based access control (RBAC) mechanisms shall be implemented to restrict access to sensitive functionalities and administrative features based on user roles and permissions.
Authentication mechanisms such as multi-factor authentication (MFA) shall be available to enhance user account security.

Maintainability

Code Modularity:

The application shall adhere to modular design principles, with well-defined components and separation of concerns to ease code maintenance and extensibility.
Code documentation, comments, and coding standards shall be enforced to promote readability and maintainability.

Version Control:

Source code shall be managed using version control systems (e.g., Git), with regular commits, branching strategies, and release tagging for tracking changes and managing codebase versions.

Portability

Cross-Browser Compatibility:

The food ordering application shall be compatible with major web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge, across different operating systems anddevices.
Responsive design techniques shall be employed to ensure best display and usability on desktops, laptops, tablets, and smartphones.

Cloud Deployment:

The application shall be designed for deployment in cloud environments such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP), using cloud-native services and infrastructure for scalability and flexibility.
These non-functional requirements define the quality attributes and constraints that shape the overall performance, reliability, availability, security, maintainability, and portability of the food ordering web application, ensuring that it meets user expectations and business aims effectively.

5.MODULES

This section will outline key modules of our Node.Js food ordering application, accompanied by code snippets.

1.Menu page:

```
const mongoose = require('mongoose')

const Schema = mongoose.Schema

const menuSchema = new Schema({
    name: { type: String, required: true },
    image: { type: String, required: true },
    price: { type: Number, required: true },
    size: { type: String, required: true }
})


module.exports = mongoose.model('Menu', menuSchema)
```

2:Order.js:

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

const orderSchema = new Schema({
    customerId: {
                type: mongoose.Schema.Types.ObjectId,
                ref: 'User',
                required: true
                },
    items: { type: Object, required: true },
    phone: { type: String, required: true},
    address: { type: String, required: true},
    paymentType: { type: String, default: 'COD'},
    status: { type: String, default: 'order_placed'},
```

```
}, { timestamps: true })

module.exports = mongoose.model('Order', orderSchema)
```

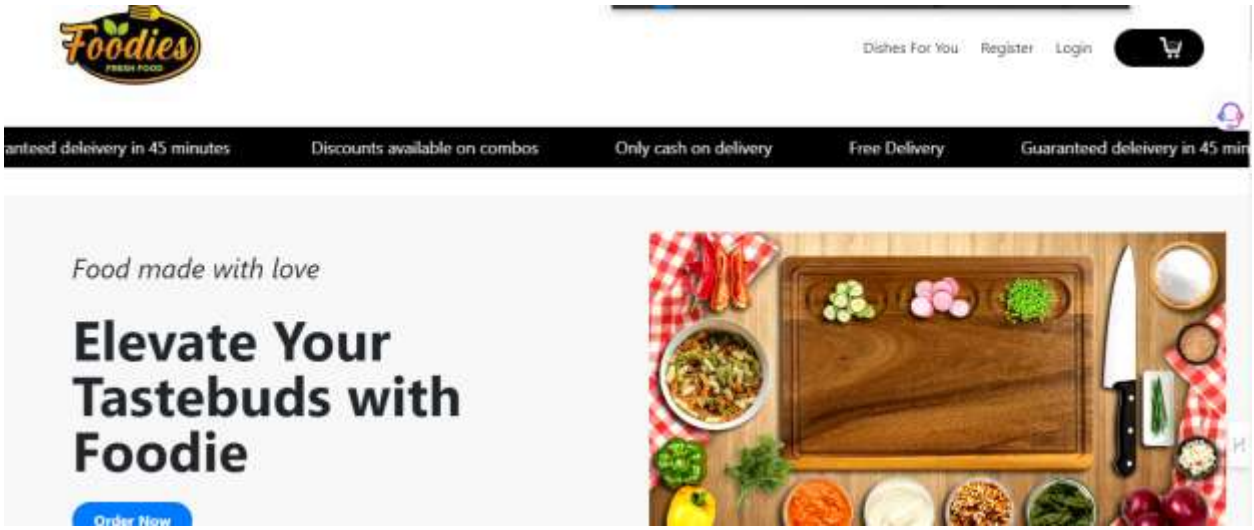
3:Admin Page:

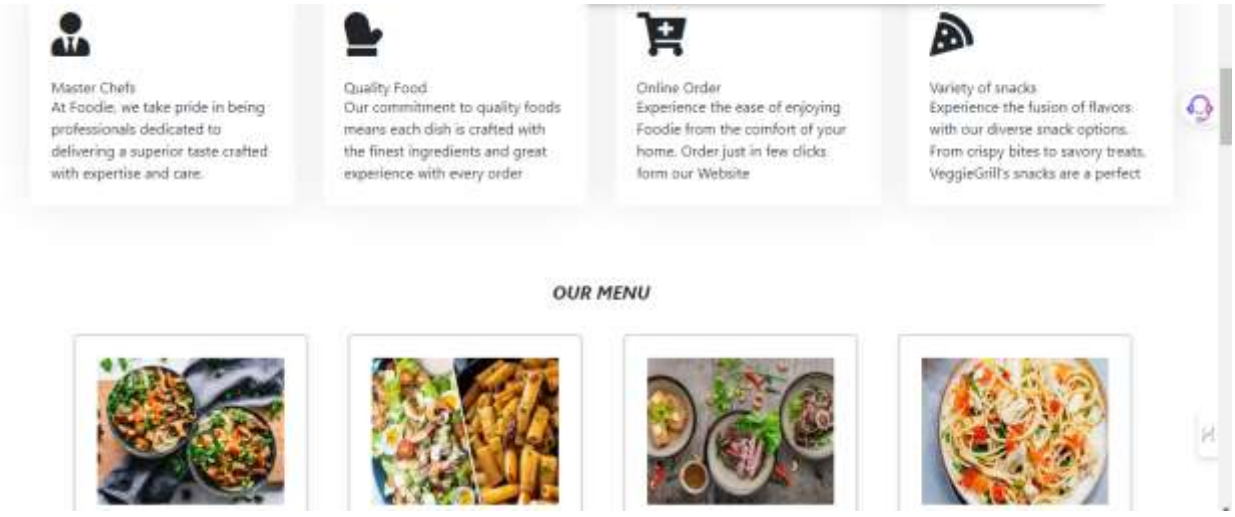
```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

const userSchema = new Schema({
    name: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    role: { type: String, default: 'customer' }
}, { timestamps: true })

module.exports = mongoose.model('User', userSchema)
```

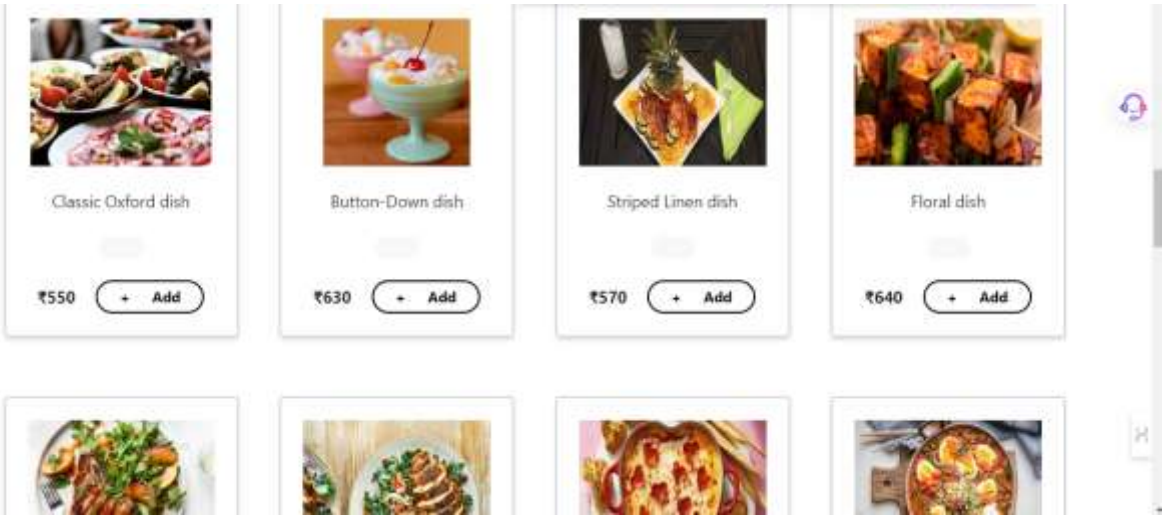
4:Order.ejs:

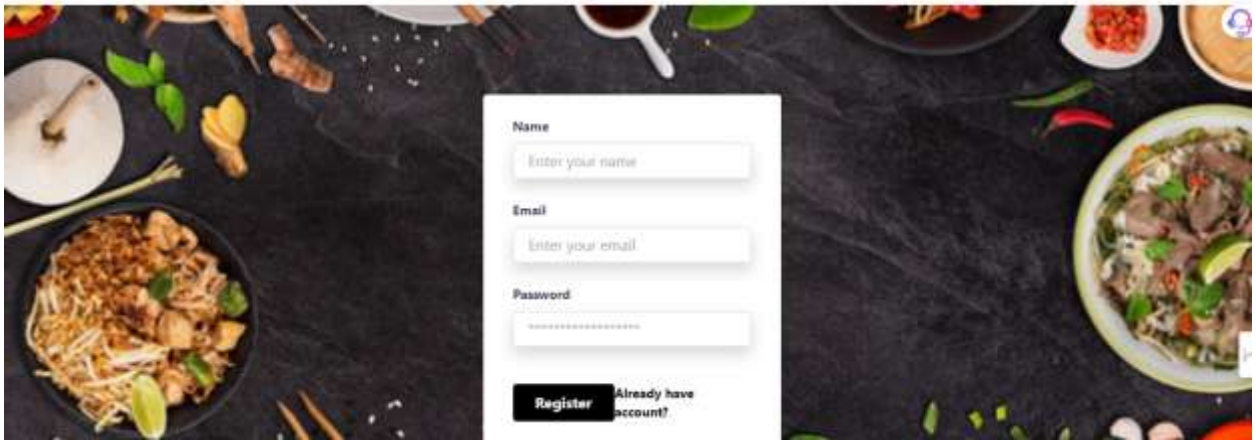
```
 <section class="orders light-section">
    <div class="container mx-auto pt-12">
        <h1 class="font-bold text-lg mb-4">All orders</h1>
        <table class="w-full table-auto bg-white">
            <thead>
                <tr>
                    <th class="px-4 py-2 text-left">Orders</th>
                    <th class="px-4 py-2 text-left">Customer</th>
                    <th class="px-4 py-2 text-left">Address</th>
                    <th class="px-4 py-2 text-left">status</th>
                    <th class="px-4 py-2 text-left">Placed at</th>
                </tr>
            </thead>
            <tbody id="orderTableBody">
            </tbody>
        </table>
    </div>
</section>
```

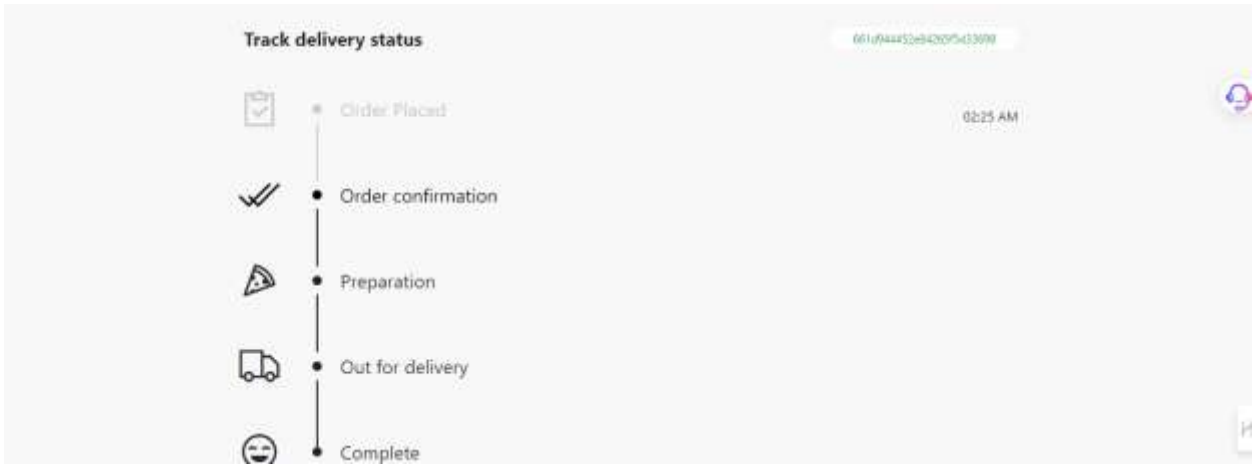
6.WEBSITE SNAPSHOTS






Classic Oxford dish
₹550
+ Add
Button-Down dish
₹630
+ Add
Striped Linen dish
₹570
+ Add
Floral dish
₹640
+ Add


Name
Enter your name
Email
Enter your email
Password
Register
Already have account?


Track delivery status
Order Placed
02:25 AM
Order confirmation
Preparation
Out for delivery
Complete

7.GITHUB LINK

Link**: https://github.com/Akshat1224/Foodies**
Hosted Website Link: https://foodies-gv9i.onrender.com

8.LINK OF REFERENCES

Link 1:https://www.w3schools.com/js/default.asp

Link 2:https://getbootstrap.com/docs/5.3/components/navbar/#nav

Link 3:https://www.npmjs.com/