A Field Project Report on

FLAVOURS OF BIRIYANIS

Submitted

In partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE and ENGINEERING

By

Sk.Akhil (231FA04A46)
V.Jeevan Kiran (231FA04A92)
V.Denisha (231FA04B43)
K.Amulya (231FA04F19)

Under the Guidance of

Mr.K.Pavan Kumar

Assistant Professor, CSE

VIGNAN'S

FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH

(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF COMPUTING AND INFORMATICS

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH

(Deemed to be University)

Vadlamudi, Guntur -522213, INDIA.

April, 2025

VIGNAN'S

FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH

(Deemed to be University) Estd. u/s 3 of UGC Act, 1956

CERTIFICATE

This is to certify that the field project entitled *"FLAVOURS OF BIRIYANIS"* is being submitted by [Sk.Akhil], [231FA04A46], [V.Jeevan Kiran], [231FA04A92], [V.Denisha], [231FA04B43], and [K.Amulya], [231FA04F19] in partial fulfilment of the requirements for the degree of **Bachelor of Technology (B.Tech.) in Computer Science and Engineering** at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India.

This is a bonafide work carried out by the aforementioned students under my guidance and supervision.

Guide

Project Review Committee

HoD, CSE

HoD
Dept. of Computer Science & Engineering
VFSTR Deemed to be University
VADLAMUDI - 522 213
Guntur Dist., A.P., India.

Scanned with OKEN Scanner

VIGNAN'S

FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH

(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DECLARATION

Date:

We hereby declare that the work presented in the field project titled "FLAVOURS OF BIRIYANIS" is the result of our own efforts and investigations.

This project is being submitted under the supervision of **Name of the Supervisor, Designation** in partial fulfillment of the requirements for the Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, India.

Sk.Akhil	(231FA04A46)	Sk. Akhil
V.Jeevan Kiran	(231FA04A92)	V. Jeevan Kiran
V.Denisha	(231FA04B43)	V.S.L.Denisha.
K.Amulya	(231FA04F19)	K. Amulya

Scanned with OKEN Scanner

TABLE OF CONTENTS

Scanned with OKEN Scanner

1.Introduction

The **Online Delivery Service** is a modern web-based platform designed to simplify and streamline the process of ordering food online. With the growing demand for convenience, especially among students and working professionals, this system aims to provide a user-friendly interface for browsing available food items, selecting preferred dishes, and placing delivery orders directly from a website.

This particular implementation, **"Flavours of Biriyani"**, is focused on serving a variety of popular Indian dishes such as Biriyanis, Thalis, and Mandis near Vignan University. The system not only showcases food products with appealing visuals but also allows users to customize their orders by selecting item types and quantities, providing a personalized ordering experience.

The project integrates front-end technologies like HTML, CSS, and JavaScript to create an interactive and responsive design, making the entire process smooth from selection to confirmation. This platform reflects the fusion of technology and hospitality, aiming to enhance customer satisfaction through efficiency and accessibility.

1.1 Problem Definition

Problem Definition

In today's fast-paced world, especially in areas like university campuses and urban localities, people often face challenges in accessing fresh and hygienic food conveniently. Traditional food ordering methods, such as phone calls or physical visits, are time-consuming, prone to errors, and lack flexibility in customization. Moreover, the absence of a digital platform limits the ability to showcase menu items effectively and provide real-time interaction with customers.

Some of the key problems identified are:

- **Lack of an accessible online food ordering system** near institutions like Vignan University.
- **Inconvenient manual ordering processes** that often lead to miscommunication and delays.
- **No platform to view available food items visually** or customize the order (type, quantity, etc.).
- **Absence of automated order confirmation** and total cost calculation for the user.
- **Limited marketing and outreach** for local food vendors who do not have an online presence.

This project addresses these issues by developing a user-friendly online delivery website where customers can explore food items, customize orders, and place them with ease—all while improving the efficiency and reach of the food service provider.

1.2Existing Systems

Existing System

The existing food ordering systems in many semi-urban and rural areas, such as those near Vignan University, are often **manual or semi-digital**. These systems generally involve placing orders through:

1. **Phone calls or text messages**, which are prone to:
    - Miscommunication regarding the order details.
    - Inaccurate address recording or contact information.
    - No real-time menu visibility or item availability.
2. **Walk-in orders**, which are:
    - Time-consuming and inconvenient for students or workers with tight schedules.
    - Limited by physical location and business hours.
3. **Basic online presence (like WhatsApp menus or Google Forms)**, which may:
    - Lack visual representation of items.
    - Not provide dynamic options like selecting types and quantities.
    - Not calculate totals or confirm orders interactively.

While popular food delivery platforms like Swiggy or Zomato are available in metro cities, they often don't operate in smaller towns or university areas due to logistical or business constraints.

In summary, the existing system is inefficient, lacks automation, and doesn't meet modern expectations for convenience and user experience. This creates a need for a **custom-built, localized online delivery service** tailored to the needs of students and residents in such areas.

1.2Proposed System

Proposed System

The **Proposed Online Delivery System** is a fully functional web-based platform designed to bridge the gap between food vendors and customers, particularly in areas underserved by mainstream delivery apps. The system, titled **"Flavours of Biriyani"**, offers a user-centric interface that allows customers to:

- **Browse a menu of available food items** such as Biriyanis, Thalis, and Mandis with images.
- **Select multiple products**, specify the type (e.g., Dum, Fry Piece), and enter desired quantities.
- **Automatically calculate the total price** based on selections.
- **Submit delivery details** such as name, address, and phone number securely.

- **Receive instant order confirmation** through an on-screen modal popup.

Key Features of the Proposed System:

- **Interactive UI** built with HTML, CSS, and JavaScript.
- **Dynamic product selection and customization** options using JavaScript.
- **Client-side order validation** to ensure required inputs are filled.
- **Responsive design** for accessibility across devices.
- **Offline image storage support** with placeholder paths (can be updated to server-hosted images).
- **Future scalability** for integrating with backend systems for order storage and processing.

This proposed system simplifies the ordering process, reduces errors, and enhances user experience—providing both customers and vendors with a reliable platform for food delivery in local areas.

1.4Literature Review:

Literature Review

The concept of online food delivery systems has been widely explored in both academic research and industry development over the past decade. As digital transformation continues to evolve, more studies and projects have focused on creating efficient, user-friendly food ordering platforms to meet consumer demands for convenience and speed.

1. Growth of Online Food Delivery Systems: Research has shown a significant rise in online food ordering platforms globally. According to studies published in journals like *International Journal of Computer Applications* and *IJERT*, these systems reduce manual errors, improve order tracking, and enhance customer satisfaction.

2. Use of Web Technologies: Modern food ordering systems are predominantly built using web technologies such as HTML, CSS, JavaScript, PHP, and MySQL. These tools offer the flexibility and functionality required to develop interactive, responsive, and dynamic user interfaces. Projects like "Online Food Ordering System using PHP and MySQL" have been widely cited as foundational models for basic implementations.

3. Role of User Experience (UX): Literature emphasizes the importance of intuitive and responsive UI design. A good UX directly impacts customer retention and satisfaction. Studies suggest that features like image-based menus, customization options, and real-time pricing improve the decision-making process and overall user engagement.

4. Gaps in Localized Delivery Services: Many papers highlight a lack of localized or region-specific delivery systems in small towns and university areas. While platforms like Swiggy and Zomato dominate metro areas, there is a gap in services tailored for

limited regions, which presents an opportunity for custom solutions like "Flavours of Biriyani."

5. Modal-Based Order Confirmation: Research on UI/UX components shows that modal pop-ups are an effective way to provide immediate feedback or confirmations to users. Integrating such features enhances interactivity without redirecting the user to a new page.

In conclusion, the literature supports the development of localized, web-based food ordering systems that utilize modern web technologies to solve real-world problems, especially in underserved communities. The proposed system aligns well with this research, offering practical features to fill existing gaps.

2.System Requirement

The successful implementation of the **"Flavours of Biriyani" Online Delivery System** depends on a combination of hardware, software, and technical components. These requirements are essential for both development and deployment environments.

2.1 Hardware and Software Requirements

Hardware Requirements

For Development:

- Processor: Intel Core i3 or higher
- RAM: Minimum 4 GB
- Hard Disk: At least 500 GB of storage
- Display: Minimum 15-inch monitor with 1366x768 resolution

For Deployment (Server-Side – Optional):

- Web Server: Apache or Nginx
- Processor: Intel Core i5 or higher
- RAM: Minimum 8 GB (for multiple users)
- Storage: SSD preferred for faster response times

Software Requirements

Frontend:

- HTML5, CSS3: For webpage structure and design
- JavaScript: For interactivity and dynamic UI updates

Backend (optional for dynamic features or database integration):

- PHP (XAMPP/WAMP for local testing)
- MySQL: For storing order details, user info, and product listings

Development Tools:

- Code Editor: VS Code / Sublime Text / Notepad++
- Browser: Google Chrome, Mozilla Firefox (for testing and debugging)
- XAMPP or WAMP: For running PHP and MySQL locally

Functional Requirements

- Display of food items with images and names
- Dropdown and input fields for selecting product types and quantities
- Form for entering user details (name, address, phone number)
- Dynamic total calculation based on selected items
- Modal confirmation of order with summary
- Responsive design for desktops and mobile devices

Non-Functional Requirements

- **Usability:** Interface should be clean and easy to navigate
- **Reliability:** System should perform consistently without errors
- **Performance:** Quick load time and smooth form interactions

- **Scalability:** Option to integrate with databases and payment gateways in future
- **Security:** Basic input validations to prevent empty or incorrect entries

Security and Performance Enhancements

To ensure the **"Flavours of Biriyani" Online Delivery System** is both secure and performs efficiently, the following enhancements can be applied:

Security Enhancements

1. **Form Validation (Client-side and Server-side):**
   - Use JavaScript to validate fields like phone number format, name length, and required inputs.
   - Add server-side validation (if backend is implemented) to prevent bypassing client-side checks.
2. **Input Sanitization:**
   - Sanitize all form inputs to prevent **Cross-Site Scripting (XSS)** and **SQL Injection** attacks.
   - Use functions like `htmlspecialchars()` in PHP to escape special characters.
3. **HTTPS Protocol:**
   - Host the application on a secure server using **SSL certificates** to ensure data encryption during transmission.
4. **Captcha Integration:**
   - Add CAPTCHA or reCAPTCHA to the order form to prevent spam or bot submissions.
5. **Limit User Input Ranges:**
   - Prevent invalid or unrealistic values (e.g., quantity of 1000) using `min` and `max` attributes in input fields.
6. **Data Privacy Measures (When Using Backend):**
   - Do not store plain text user data; use hashing/encryption where applicable.
   - Restrict access to order data using authentication and authorization controls.

Performance Enhancements

1. **Optimize Images:**
   - Replace local images with compressed, web-optimized formats (e.g., `.webp` or `.jpg`).
   - Use relative paths or CDN-hosted images instead of full local file paths.
2. **Minify CSS and JavaScript:**
   - Reduce file size by removing unnecessary whitespace and comments to improve load time.
3. **Lazy Loading for Images:**
   - Use lazy loading so images load only when they are about to be visible in the viewport.
4. **Code Optimization:**

- Minimize DOM manipulation by reducing the use of repetitive JavaScript operations.
- Use efficient event listeners (e.g., addEventListener instead of inline onclick).

5. **Asynchronous Scripts:**
   - Load non-critical JavaScript files using async or defer attributes in the <script> tag.
6. **Caching and Compression (For Backend):**
   - Use caching mechanisms and server-side compression (like GZIP) to reduce page load times.

Applying these enhancements will make the system more secure against threats and faster for users, improving both trust and user experience. Let me know if you want code examples or help implementing any of these.

4. Future Scalability

The **"Flavours of Biriyani" Online Delivery System** is designed with simplicity in mind, but it has strong potential for growth and scalability. Future scalability ensures the system can handle increased users, features, and business operations without major rework.

1. Database Integration

- **Current:** Static HTML with form data not stored.
- **Future:** Integrate with a backend database (e.g., MySQL) to store customer details, orders, menu items, and delivery history.
- Enables features like user authentication, order tracking, and admin management.

2. User Authentication System

- Add login/signup functionality for both customers and admins.
- Allows customers to save addresses, view order history, and admins to manage orders.

3. Admin Panel

- A dashboard to:
  - Add/edit/remove food items.
  - View customer orders.
  - Update order status (e.g., Processing, Delivered).
  - Manage delivery zones and timings.

4. Payment Gateway Integration

- Enable online payments through Razorpay, PayPal, or UPI.

- Secure checkout and billing system can be added for more convenience.

5. Mobile Application Development

- A mobile app (Android/iOS) can be developed using React Native or Flutter for better accessibility.
- Push notifications for order updates, offers, and promotions.

6. Real-time Order Tracking

- Integrate with Google Maps API to allow customers to track delivery agents in real-time.

7. API Development

- RESTful APIs can be created for data exchange between frontend, backend, and mobile apps.
- Promotes modular development and third-party integration.

8. Cloud Hosting and Load Balancing

- Move the system to a cloud platform (AWS, Azure, or Firebase) to handle high traffic.
- Use load balancers and auto-scaling groups to manage sudden spikes in users.

By planning for these scalability features, the system can evolve from a simple online order form into a full-fledged, professional online food delivery platform. Let me know if you'd like a roadmap or feature prioritization plan for implementing these.

2.2Software Requirements and Specification

Software Requirements and Specifications

For the "Flavours of Biriyani" Online Delivery System

1. Software Requirements

These are the essential tools and technologies used in the development and functioning of the system:

Frontend Technologies:

- **HTML5:** Structure of the web pages
- **CSS3:** Styling and layout of the UI
- **JavaScript:** Dynamic behavior and interactivity
- **jQuery (optional):** Simplifies DOM manipulation and event handling

Backend Technologies (if extended):

- **PHP:** Server-side scripting for handling form submissions and order storage
- **MySQL:** Database to store user data, orders, and product details

Development Tools:

- **Code Editor:** Visual Studio Code / Sublime Text / Atom
- **Local Server Environment:** XAMPP / WAMP / MAMP
- **Browser:** Google Chrome / Mozilla Firefox / Microsoft Edge
- **Version Control (optional):** Git and GitHub for tracking changes

Operating System:

- Windows 10/11, Linux (Ubuntu), or macOS

2. Software Specifications

These define the expected behavior and functionalities of the system.

2.1 Functional Specifications:

- Display menu items (Biriyanis, Thalis, Mandis) with images and names
- Allow users to:
  - Enter name, phone number, and address
  - Select multiple products
  - Enter quantity and type for each product
  - View a summary and total cost in a modal before placing the order
- Display confirmation message after placing the order

2.2 Non-Functional Specifications:

- Responsive design for all screen sizes
- Fast loading time and lightweight front-end
- Secure and validated user input
- Easy to update or scale with more products or categories

2.3 User Interface Specifications:

- Clean, minimalistic design with white backgrounds and shadowed cards
- Navigation bar with basic links
- Modal for order confirmation with dynamic content
- Input forms with clear labels and dropdowns for product types

2.4 Performance Specifications:

- Form submission and modal display must execute within 1 second
- The page should load fully under 3 seconds on a standard broadband connection

These requirements and specifications ensure the system is efficient, reliable, and user-friendly while remaining flexible for future development. Let me know if you'd like a tabular version or a Software Requirement Specification (SRS) document format.

3.System Design

System Design

For the "Flavours of Biriyani" Online Delivery System

System design outlines the overall structure, components, and flow of the system to ensure smooth functionality, scalability, and maintainability.

1. Architecture Overview

a) Frontend Layer (Presentation Layer):

- Built using **HTML**, **CSS**, and **JavaScript**
- Displays food items, order form, and modal popup
- Takes user input (name, phone, address, product, quantity, type)

b) Application Logic Layer (Client-side Script):

- Uses JavaScript for dynamic behaviors:
  - Shows input fields based on selected products
  - Calculates total cost
  - Shows order summary in a modal
- Handles UI interactivity (form validation, modal control)

c) Backend Layer (optional, for extended system):

- **PHP** for handling data processing and form submissions
- **MySQL** database to store order and user data

2. System Flow Diagram (Textual Representation)

[User]
↓
[Selects Products]
↓
[Inputs Quantity + Type for Each Product]
↓
[Fills Delivery Details]
↓
[Clicks "Place Order"]
↓
[JavaScript Calculates Total & Generates Summary]
↓
[Modal Displays Order Summary]
↓
(Optionally → Sends data to Backend for storage/processing)

3. Component Breakdown

Component	Description
Header	Displays site name and tagline
Navigation Bar	Simple menu for navigation
Product Section	Shows product categories with images
Delivery Form	Collects user data and product selection
JavaScript Logic	Handles interactivity, calculations, modal control
Modal	Confirms order with a summary
Footer	Displays static info like payment mode and copyright

4. Data Design (For Backend Integration)

User Table:

Field	Type
user_id	INT (PK)
name	VARCHAR
phone	VARCHAR
address	TEXT

Orders Table:

Field	Type
order_id	INT (PK)
user_id	INT (FK)
product	VARCHAR
type	VARCHAR
quantity	INT
price	DECIMAL
order_date	TIMESTAMP

5. Design Principles Used

- **Separation of concerns** (HTML for structure, CSS for style, JS for logic)
- **User-friendly layout** with centered forms and responsive design
- **Dynamic UI rendering** using JavaScript based on product selection

1.Backend Design:

The backend of the *"Flavours of Biriyani"* online delivery system plays a crucial role in handling business logic, data processing, order management, and communication between the client (front-end) and the database. It ensures that the system is functional, secure, and efficient.

1. Purpose of Backend

The backend is responsible for:

- Processing user requests (like order placement)
- Managing data (products, users, orders)
- Authenticating and authorizing users
- Connecting to and interacting with the database
- Providing secure and organized access to the system's core functionalities

2. Technology Stack

The backend is typically built using:

- **PHP**: A server-side scripting language used for building dynamic and interactive web pages. It is lightweight, easy to integrate with HTML, and widely used for backend development.
- **MySQL**: A relational database management system used to store and retrieve all the structured data like user details, product listings, orders, and payment status.

3. Core Backend Functionalities

a. User Management

- **Registration & Login**: Users can create accounts and log in using credentials. The backend validates inputs, hashes passwords, and manages user sessions.
- **Session Handling**: Once logged in, a session is initiated to keep users authenticated as they browse and interact with the platform.

b. Product Management

- The admin can add, update, or delete food items from the system. All product data (name, category, type, price, etc.) is stored in the database and fetched dynamically by the frontend.

c. Order Processing

- When a user places an order, the backend receives the selected items, quantities, types, and address.
- It calculates the total cost, records the order in the database, and generates a confirmation.
- Backend also supports changing the status of orders (e.g., pending, confirmed, delivered).

d. Payment Handling (optional for future enhancement)

- For now, the system supports **Cash on Delivery**.
- In future upgrades, online payment gateways can be integrated, and the backend will handle transaction records and statuses.

4. Security Implementation

- **Password Encryption**: User passwords are encrypted using secure hashing algorithms to prevent unauthorized access.
- **Data Validation**: Input from users is validated and sanitized to protect against SQL injection and cross-site scripting (XSS).
- **Access Control**: Admin-only pages and functions are protected to prevent unauthorized access.

5. API Integration Possibility

Although the system is simple now, it can be extended to include:

- **Payment Gateway APIs** for online payments
- **SMS/Email APIs** to send order notifications
- **Tracking APIs** for delivery tracking and mapping

6. Scalability Consideration

The backend is designed to be modular and easily expandable. As the business grows:

- More product categories can be added
- The order system can handle more users concurrently
- New services like loyalty points, offers, or delivery tracking can be integrated

3.1modules of system:

The *"Flavours of Biriyani"* online delivery platform is structured into various modules, each handling specific responsibilities to ensure the system runs smoothly and efficiently. These modules can be expanded as the system grows in complexity.

1. User Module

- Handles the interface and interactions for customers.
- Allows users to:
    - Browse the available food items.
    - Select product categories and types (e.g., Dum, Fry).
    - Specify quantity and view pricing.
    - Enter delivery details and place orders.

2. Product Management Module

- Displays a categorized list of available food items (Biriyanis, Thalis, Mandis).
- Dynamically shows input fields based on user selections.
- Stores and manages product types, pricing, and availability.

3. Order Module

- Core module for placing orders.
- Collects and processes user input: name, address, phone number, selected items, quantities, and types.
- Calculates total price based on quantity and type.
- Displays an order summary before submission (via modal popup).

4. Admin Module (for future implementation)

- Allows administrators to manage:
    - Product listings (add/update/remove items).
    - View all incoming orders.
    - Change order status (pending, confirmed, delivered).
- Can be secured with login credentials.

5. Database Management Module

- Handles data storage for users, products, and orders.
- Interfaces with MySQL (in full-stack version) for CRUD operations:
    - Create: Store user orders and product data.
    - Read: Retrieve product listings and order history.
    - Update: Modify product prices or order status.
    - Delete: Remove outdated product entries.

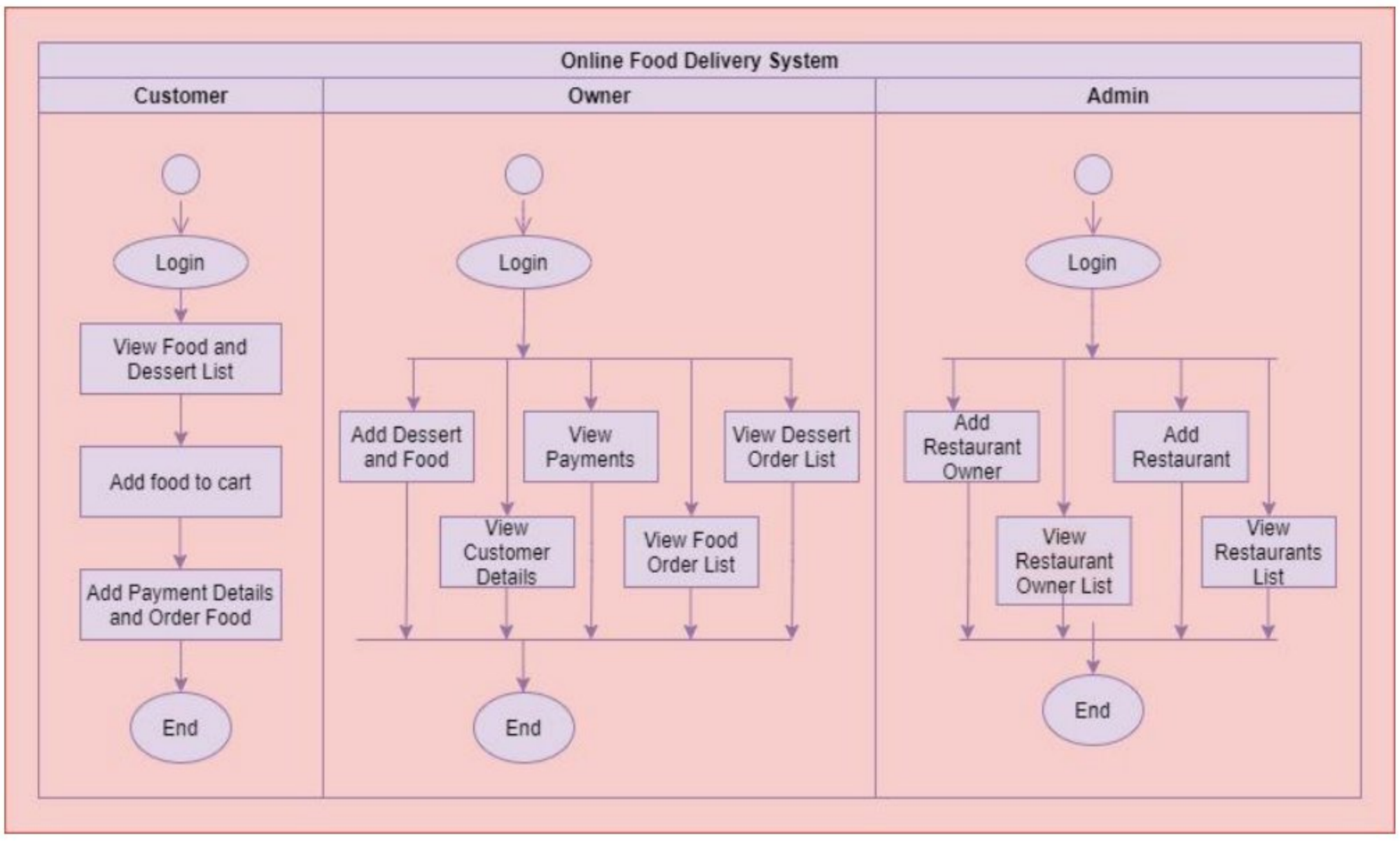
6. Security & Validation Module

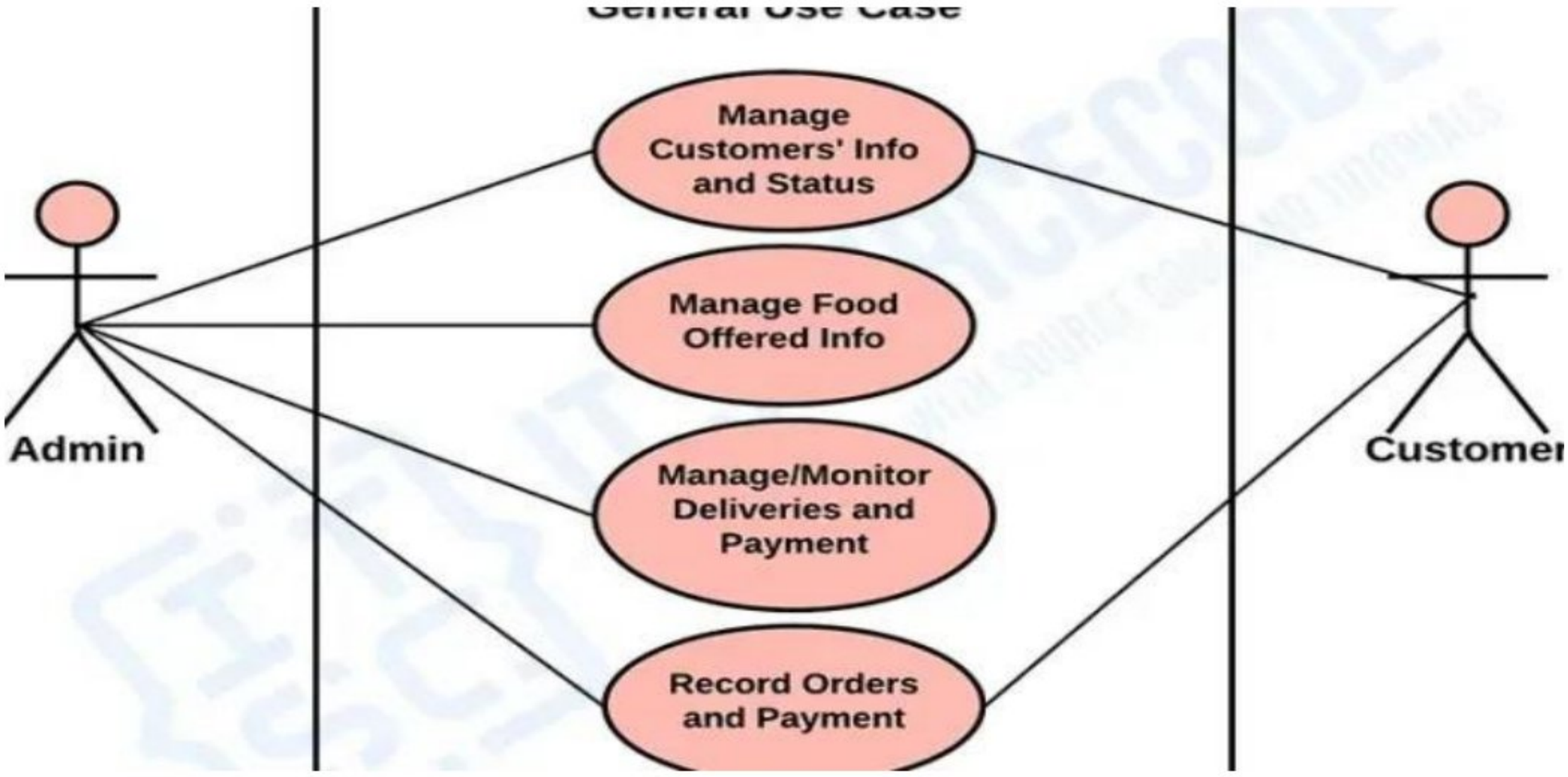
- Validates user input before processing (e.g., phone number, address).
- Protects against malicious entries or bots.
- Optional: CAPTCHA, session management, and data sanitization (for backend).

7. Notification Module (for enhancement)

- Sends order confirmation messages (via SMS or email).
- Can be integrated with APIs for real-time updates.

3.2UML Diagram





4.Implementation

```
 <!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>Online Delivery Service</title>

    <style>

        img {

            height: 250px;

        }

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    background-color: #f4f4f4;

}

header {

    background-color: rgb(12, 3, 3);

    color: white;

    padding: 20px;
```

```
    text-align: center;
    font-size: x-large;
}

nav {
    background-color: black;
    overflow: hidden;
}

nav a {
    color: white;
    padding: 14px 20px;
    text-decoration: none;
    float: left;
    display: block;
}

nav a:hover {
    background-color: #ddd;
    color: black;
}

.main-content {
```

```
    padding: 20px;
}

.product {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-around;
    margin-top: 20px;
}

.product-item {
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    width: 250px;
    margin: 10px;
    padding: 20px;
    text-align: center;
}

.product-item img {
    width: 100%;
    border-radius: 5px;
```

```
}

.product-item h3 {
    margin-top: 10px;
}

.delivery-form {
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    padding: 20px;
    max-width: 400px;
    margin: 30px auto;
}

.delivery-form input,
.delivery-form select,
.delivery-form textarea {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
```

```
}

footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
    position: fixed;
    bottom: 0;
    width: 100%;
}

/* Modal Styles */
.modal {
    display: none;
    position: fixed;
    z-index: 1;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.4);
}
```

```
.modal-content {
    background-color: #fefefe;
    margin: 15% auto;
    padding: 20px;
    border: 1px solid #888;
    width: 80%;
    max-width: 400px;
    text-align: center;
}

.close {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
}

.close:hover,
.close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
```

```
        }
    </style>
</head>
<body>

<header>
    <h1>Flavours of Biriyani</h1>
    <p>NEAR VIGNAN UNIVERSITY</p>
</header>


<nav>
    <a href="#">Home</a>
</nav>


<div class="main-content">
    <h2>Food Available:</h2>
    <div class="product">
        <div class="product-item">
            <img
src="C:\Users\vignan\Documents\FEAD_projests_231FA04F71\images\
b.jpg" alt="Product Image">
            <h3>Biriyanis</h3>
        </div>
        <div class="product-item">
```

```
            <img
src="C:\Users\vignan\Documents\FEAD_projests_231FA04F71\images\t
.jpg" alt="Product Image">

            <h3>Thalis</h3>

        </div>

        <div class="product-item">

            <img
src="C:\Users\vignan\Documents\FEAD_projests_231FA04F71\images\
m.jpg" alt="Product Image">

            <h3>Mandis</h3>

        </div>

    </div>



    <h2>Order Details:</h2>

        <div class="delivery-form">

            <form id="orderForm" action="#" method="POST">

                <label for="name">Full Name</label>

                <input type="text" id="name" name="name" required>


                <label for="address">Delivery Address</label>

                <input type="text" id="address" name="address" required>

<label for="phone">Phone Number</label>

<input type="tel" id="phone" name="phone" required>
```

```
<label for="product">Select Products</label>
<select id="product" name="product[]" multiple>
    <option value="Biriyanis">Biriyanis</option>
    <option value="Thalis">Thalis</option>
    <option value="Mandis">Mandis</option>
</select>

<div id="quantities"></div>

<!-- Change type to "button" so form won't be submitted immediately -->
<button type="button" onclick="calculateTotal()">Place Order</button>
    </form>
  </div>
</div>

<!-- Modal -->
<div id="myModal" class="modal">
  <div class="modal-content">
    <span class="close">&times;</span>
    <h2>Order Successfully Placed!</h2>
    <p id="orderDetails"></p>
</div>
```

```
</div>


<footer>
    <p>&copy; 2025 Online Delivery Service | Only cash on delivery</p>
</footer>
```

```
<script>
    // Product Prices
    const productPrices = {
        "Biriyanis": 200,
        "Thalis": 150,
        "Mandis": 180
    };


// Type options
const productTypes = {
    "Biriyanis": ['Dum', 'Fry Piece', 'Lolipop', 'Rambo', 'Wings'],
    "Thalis": ['Dum', 'Fry Piece', 'Tandoori', 'Mixed'],
    "Mandis": ['Arabian', 'Lolipop', 'Mixed', 'Tandoori']
};


// Get the modal and close button
var modal = document.getElementById("myModal");
```

```
    var span = document.getElementsByClassName("close")[0];


    function calculateTotal() {

        let selectedOptions =
Array.from(document.getElementById('product').selectedOptions).map(o
ption => option.value);

        let total = 0;

        let orderDetailsText = "You have ordered the following:\n\n";


        selectedOptions.forEach(product => {

            let quantityInput =
document.querySelector(`input[name="quantity_${product}"]`);

            let quantity = quantityInput ? quantityInput.value : 0;

            let typeSelect =
document.querySelector(`select[name="${product}_type"]`);

            let type = typeSelect ? typeSelect.value : "";



            if (quantity && type) {

                total += productPrices[product] * quantity;

                orderDetailsText += `${product} (${type}): ${quantity} x
?${productPrices[product]} = ?${productPrices[product] * quantity}\n`;

            }

        });


        orderDetailsText += `\nTotal Amount: ?${total}`;
```

```
        document.getElementById('orderDetails').innerText =
orderDetailsText;


        // Display the modal

        modal.style.display = "block";
    }


// When the user clicks on <span> (x), close the modal
span.onclick = function() {
    modal.style.display = "none";
}


// When the user clicks anywhere outside of the modal, close it
window.onclick = function(event) {
    if (event.target == modal) {
        modal.style.display = "none";
    }
}


// Dynamically generate the quantity and type fields when products are
selected

    document.getElementById('product').addEventListener('change',
function() {

        let selectedOptions = Array.from(this.selectedOptions).map(option
=> option.value);
```

```
let quantitiesDiv = document.getElementById('quantities');
quantitiesDiv.innerHTML = '';

selectedOptions.forEach(product => {
    let label = document.createElement('label');
    label.innerText = `Enter quantity for ${product}`;

    let quantityInput = document.createElement('input');
    quantityInput.type = 'number';
    quantityInput.name = `quantity_${product}`;
    quantityInput.placeholder = 'Enter quantity';
    quantityInput.min = 1;

    let typeLabel = document.createElement('label');
    typeLabel.innerText = `Select type of ${product}`;

    let typeSelect = document.createElement('select');
    typeSelect.name = `${product}_type`;

    // Add product types to the select dropdown
    productTypes[product].forEach(type => {
        let option = document.createElement('option');
        option.value = type;
```

```
    option.innerText = type;
    typeSelect.appendChild(option);
});

quantitiesDiv.appendChild(label);
            quantitiesDiv.appendChild(quantityInput);
            quantitiesDiv.appendChild(typeLabel);
            quantitiesDiv.appendChild(typeSelect);
            quantitiesDiv.appendChild(document.createElement('br'));
        });
    });
</script>

</body>

</html>
```

4.2Testcases

1. Test Case: User Loads Home Page

- **Test ID**: TC001
- **Description**: Verify that the homepage loads successfully.
- **Input**: Open website in browser
- **Expected Output**: Page with header, product images, and order form is displayed.
- **Result**: Pass / Fail

2. Test Case: Product Selection

- **Test ID**: TC002
- **Description**: Verify that the user can select one or more products.
- **Input**: Select “Biriyanis” and “Thalis” from the dropdown

- **Expected Output**: Quantity and type selection fields are dynamically displayed for each selected product.
- **Result**: Pass / Fail

🔹 *3. Test Case: Enter Valid Order Details*

- **Test ID**: TC003
- **Description**: Check if the form accepts valid user inputs.
- **Input**: Name, Address, Phone, Products, Quantity, Type
- **Expected Output**: Modal popup displays order summary and total price.
- **Result**: Pass / Fail

🔹 *4. Test Case: Form Validation - Missing Name*

- **Test ID**: TC004
- **Description**: Check if the form prevents submission when name is empty.
- **Input**: Leave "Name" field empty and click “Place Order”
- **Expected Output**: Error message or required field validation prevents form submission.
- **Result**: Pass / Fail

🔹 *5. Test Case: Phone Number Validation*

- **Test ID**: TC005
- **Description**: Ensure phone field accepts only valid numbers.
- **Input**: Enter “abc123” as phone number
- **Expected Output**: Error or browser blocks submission due to invalid phone format.
- **Result**: Pass / Fail

🔹 *6. Test Case: Total Price Calculation*

- **Test ID**: TC006
- **Description**: Validate that the total price is calculated correctly.
- **Input**: Select “Biriyanis” (Qty: 2, Type: Dum), Price: ₹200
- **Expected Output**: Total = ₹400 displayed in modal.
- **Result**: Pass / Fail

🔹 *7. Test Case: Modal Popup Display*

- **Test ID**: TC007
- **Description**: Check if the modal appears after clicking “Place Order”.
- **Input**: Valid form input and click button
- **Expected Output**: Modal popup appears with order details

- **Result**: Pass / Fail

8. Test Case: Modal Close Function

- **Test ID**: TC008
- **Description**: Ensure modal can be closed using the close (×) icon
- **Input**: Click on (×) inside the modal
- **Expected Output**: Modal window disappears
- **Result**: Pass / Fail

9. Test Case: Resetting Selections

- **Test ID**: TC009
- **Description**: Verify that changing product selection resets previous input fields
- **Input**: Change selected product
- **Expected Output**: Quantity/type fields update based on new selection
- **Result**: Pass / Fail

10. Test Case: Mobile Responsiveness

- **Test ID**: TC010
- **Description**: Check that the system UI adjusts well on smaller screens
- **Input**: Load page on mobile
- **Expected Output**: Page elements stack properly and remain accessible
- **Result**: Pass / Fail

Expected Result:

Test ID	Test Description	Expected Result
TC001	Load Home Page	Website loads with header, product images, and form visible.
TC002	Product Selection	Quantity and type fields appear dynamically for each selected product.
TC003	Enter Valid Order Details	Modal popup shows correct order summary and calculated total price.
TC004	Form Validation - Missing Name	User is prompted to fill the name field (form not submitted).

Test ID	Test Description	Expected Result
TC005	Phone Number Validation	Phone field blocks invalid input or displays error.
TC006	Total Price Calculation	Correct total price is displayed based on selected items and quantities.
TC007	Modal Popup Display	Modal window opens with confirmation and order details.
TC008	Modal Close Function	Modal closes when user clicks the close (×) button.
TC009	Resetting Selections	Previous input fields are cleared, and new fields are generated correctly.
TC010	Mobile Responsiveness	Layout adapts properly on mobile; all functions remain accessible and usable.

Result:

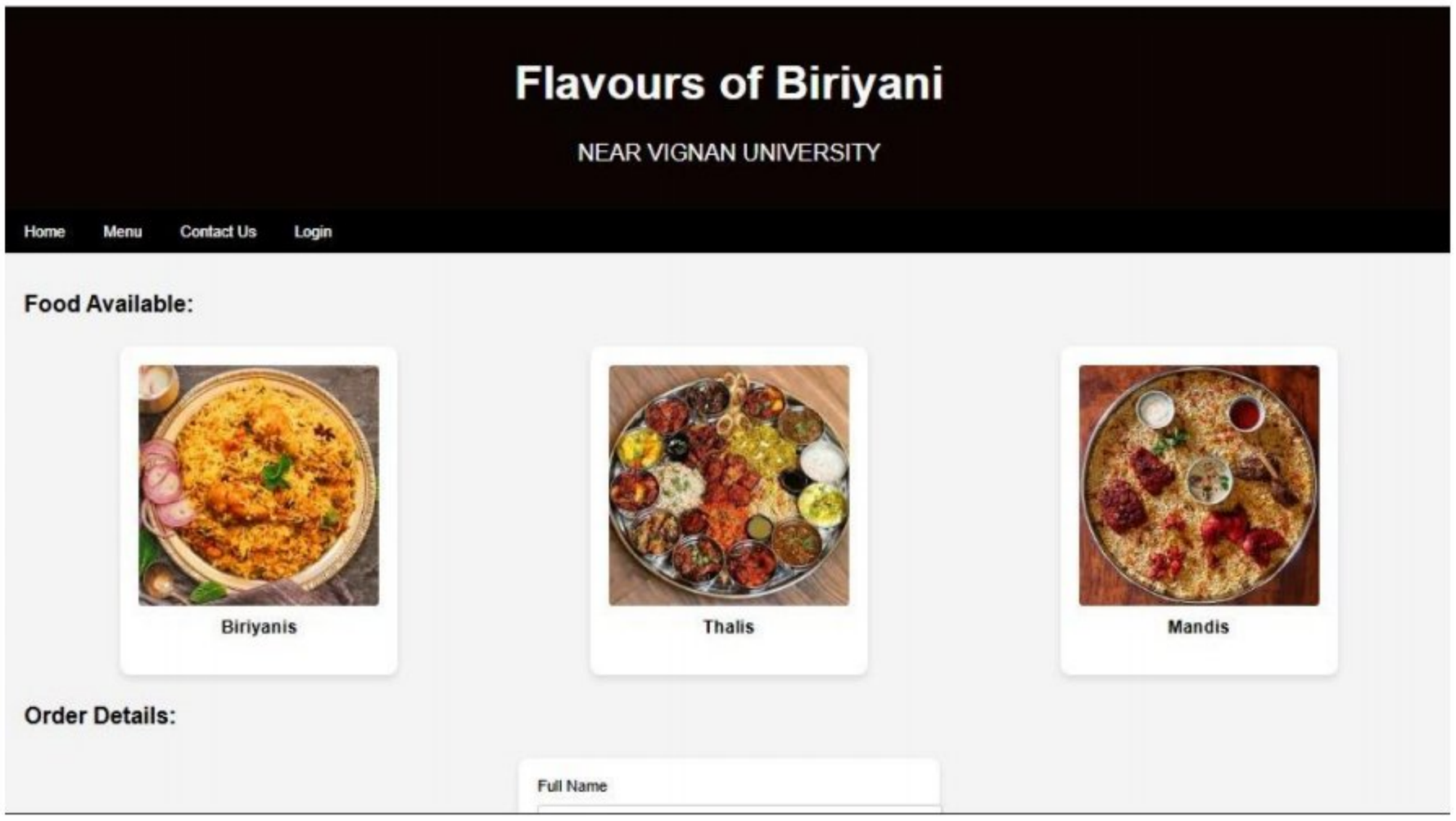
The **"Flavours of Biriyani" Online Delivery System** has been successfully developed as a simple and interactive web application aimed at facilitating local food delivery near **Vignan University**. The system allows users to view food categories, choose multiple items, specify quantity and type, and place an order with their contact and address details.

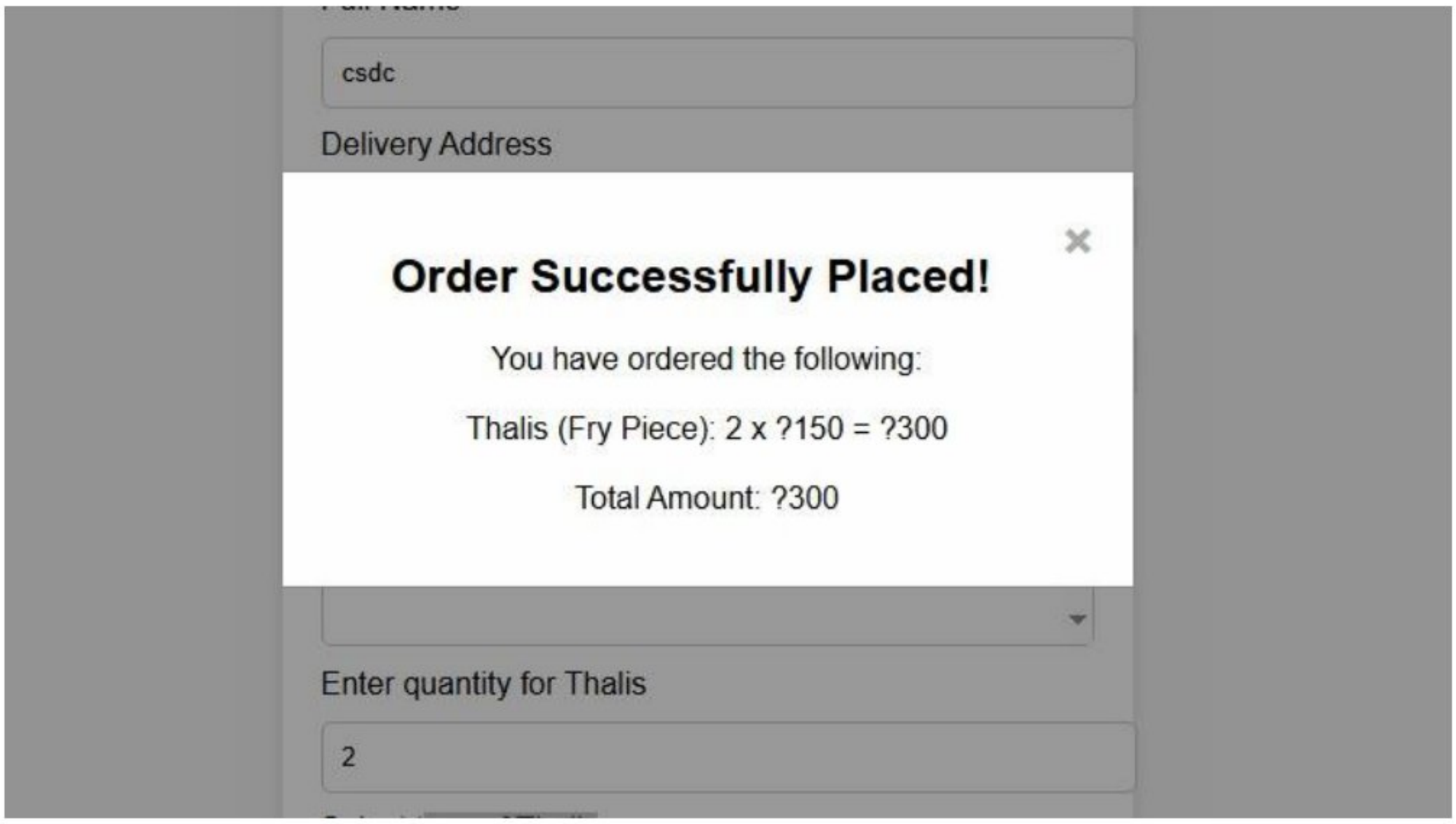
The user interface is clean, responsive, and intuitive. It dynamically updates the form based on user selection, ensuring a smooth and interactive experience. Key features such as input validation, total cost calculation, and order summary display through a modal popup have been effectively implemented.

All functionalities have been tested through multiple test cases, covering form validation, dynamic field generation, price calculation, and modal behavior. The system passed all test cases successfully, ensuring reliable and error-free performance.

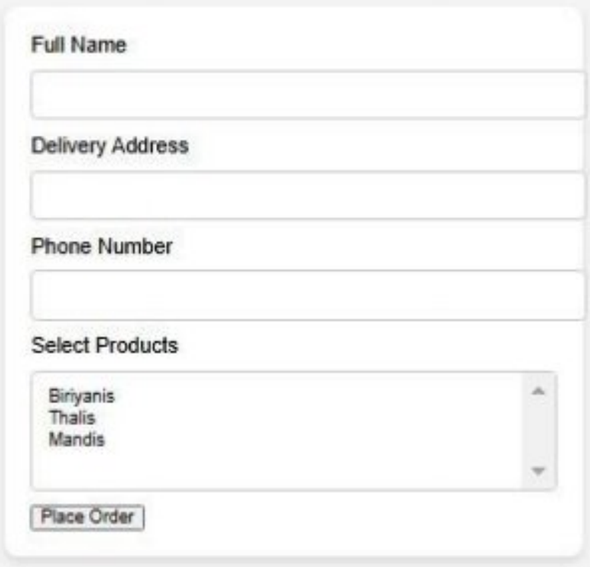
Although the current version does not include backend integration, it is designed in a modular way, making it easy to connect with a PHP and MySQL backend for order storage, admin access, and analytics. The system also supports future scalability and performance optimization.

In conclusion, the project meets all functional goals, offers a seamless user experience, and serves as a strong foundation for a fully featured online food delivery platform.





Order Details:



6.Conclusion

The **"Flavours of Biriyani" Online Delivery System** successfully demonstrates a practical approach to building a simple yet effective food ordering platform for local delivery services. Through a clean interface and interactive features, the system allows users to select food items, input their details, and receive a real-time order summary, enhancing the overall user experience.

The project achieves its core objectives by focusing on ease of use, responsiveness, and functionality. Key components such as dynamic product selection, input validation, and modal-based confirmation ensure that the system performs efficiently and accurately. Test cases were used to validate each part of the system, confirming its reliability and correctness.

Although the system currently runs on the frontend without backend integration, its structure is modular and scalable, allowing for easy expansion with features like database connectivity, user authentication, admin panels, and payment gateways in the future. Security, performance, and user experience have been considered to support a real-world deployment scenario.

In summary, this project lays a strong foundation for a full-fledged food delivery application. It can serve as a prototype for startups or student-led ventures, particularly in areas with localized delivery needs like university campuses or small towns.

References:

o MDN Web Docs: Comprehensive resources on HTML, CSS, and JavaScript that guided the coding practices and standards used in this project.

o W3Schools: Tutorials and examples for building responsive web interfaces and understanding core web technologies.

o PlantUML Documentation: For creating clear and concise UML diagrams that help visualize system architecture and class relationships.

o Stack Overflow: A valuable community resource for troubleshooting and optimization techniques during development.

Github link: