HAMK
Häme University
of Applied Sciences

A cost-effective web-based ordering system for small food businesses

Prototype development of an online ordering and delivery management system for small ethnic food businesses

Bachelor's Thesis
Degree Programme in Computer Applications
Spring 2026
Madushani Piyadasa

HAMK
Häme University
of Applied Sciences

Abstract

DP	Degree Programme in Computer Applications
Author	Madushani Piyadasa Year 2026
Subject	Design and development of a cost effective web-based ordering system for small food delivery business in Finland
Supervisors	Dr. Jawad Yasin

Rapid development in technology has changed the food delivery sector through the incorporation of online ordering systems that enhance efficiency and convenience. Nevertheless, numerous small food companies in Finland still use third-party services such as Wolt and Uber Eats, which usually come with hefty commissions and restrict business control over consumer information and processes. The purpose of this thesis is to develop an efficient web-based platform for ordering and managing deliveries for small food delivery establishments.

The main aim of the research was to create an independent system for placing orders online, whereby the use of third-party services would be minimised, while still retaining the basic features required for ordering. This was completed by creating an application using HTML, CSS, JavaScript, PHP, and MySQL programming languages in the XAMPP development platform. Rapid Application Development method was used to improve the web site.

Functionalities such as menu browsing, cart handling, order delivery, order pickup, dynamic check-out processes, order storage, and order management via the administration panel were included in the developed software. A relational database schema was introduced to provide organised and effective data storage. All functional requirements were successfully fulfilled after performing functional tests, where operations such as cart modification, delivery validation, order placement, and order viewing via the administration panel were verified.

As can be seen from the above results, the developed prototype has high potential for providing small-sized food companies with the possibility of managing their operations efficiently through a web-based ordering system. In contrast to the costly process of integrating complex commercial delivery systems, the prototype enhances the autonomy of businesses, their usability, and order management. Although important features such as payment processing, tracking, and security mechanisms have not been added, the developed solution offers a good starting point for further work.

Keywords	web-based ordering system, online food delivery system, restaurant management system, small business digitalisation, Rapid Application Development(RAD), usability design, relational database design, order management system
Pages	48 pages and appendices 2 pages

Glossary

CSS	Cascading Style Sheet
ER	Entity Relationship
GPS	Global Positioning System
HTML	HyperText Markup Language
PHP	Hypertext Preprocessor
RAD	Rapid Application Development
SQL	Structured Query Language
TAM	Technology Acceptance Model
XAMPP	Cross-Platform + Apache + MySQL + PHP + Perl
UX	User Experience

Table of Contents

Figures

Tables

Appendices

1 Introduction

The rapid evolution of digital technology has significantly transformed the functioning of firms, particularly those in the food service industry. Digital technologies that lead to increased efficiency and reduced mistakes on the part of humans are progressively replacing old-school meal ordering processes, which entailed making phone calls and using manual methods. Online ordering of meals makes it easy for customers to browse menu choices, order their meals, and even schedule delivery services (Chavan et al., 2015; Oghenekaro & Okafor, 2023)

Online meal delivery is becoming increasingly popular in Finland, especially in urban areas such as Helsinki, which has many Wolt and Uber Eats users. Some of the functions provided by digital food delivery systems include order handling, payment handling, and deliveries. Studies have indicated that such systems limit adaptability and flexibility for small organisations, causing high operating costs and commissions (Piyatissa, 2020). This is why many food businesses, particularly small ones (Oghenekaro & Okafor, 2023) , still rely on manual or semi-digital food delivery platforms that sometimes cause issues related to order handling and delays (Chavan et al., 2015).

These challenges illustrate the gap between the market’s advanced technological innovations and the practical needs of small enterprises. Despite the great capabilities offered by large-scale platforms, there is still the possibility that smaller businesses with simpler needs for cost-effective and easily managed technology may not be able to take advantage of them (Chavan et al., 2015; Piyatissa, 2020).

To bridge this gap, the main objective of this thesis is to design a web-based system of ordering and delivery management tailored to small businesses in the delivery industry (Oghenekaro & Okafor, 2023). Through a user-friendly interface, the proposed system allows customers to view menus, place orders, and provide delivery details. On the other hand, the business owner will be able to modify the menu and monitor orders and delivery (Chavan et al., 2015)

This study used an approach that combined practice and theory. Although the practical side of the research focuses on developing a working model with HTML (Hypertext Markup Language) , CSS (Cascading Style Sheet), JavaScript, PHP (Hypertext Preprocessor), and MySQL (Structured Query Language), the theoretical side covers notions such as web-

based systems, system architecture, and usability. This is because these technologies allow for the creation of dynamically generated materials and efficient data processing (Chavan et al., 2015; Oghenekaro & Okafor, 2023)

The RAD (Rapid Application Development) approach of application development, which focuses on the fast creation and iteration of prototypes, was adopted to develop the system (Oghenekaro & Okafor, 2023). This method is particularly effective in building web-based applications that require constant improvement during the development process (Chavan et al., 2015). Research suggests that RAD helps in faster system development and improves the usability of the system through iterative improvements.

Placing an order, displaying the menu, and tracking the status of orders are some of the basic elements included in the system design. In order to maintain simplicity and ensure that the system remains within the scope of a bachelor level project, advanced functionalities such as payments and GPS (Global Positioning System) monitoring have been deliberately excluded from the system (Piyatissa, 2020). This approach is in line with scholarly findings that emphasise the importance of focusing on key elements when designing systems for small businesses (Oghenekaro & Okafor, 2023).

Research questions are as follows

- How can the technical and functional specifications for a web-based ordering and delivery management system for small food delivery companies be determined?
- How can an effective and user-friendly online ordering system be created using the Rapid Application Development methodology and appropriate system architecture?
- How can small food delivery companies improve order management and reduce their dependence on external delivery platforms using a web-based ordering system?

The development of an operational system prototype with adequate documentation for architecture, database design, and usability considerations is the expected outcome of this thesis. This prototype presents the manner by which smaller food delivery companies can utilise simple technological solutions to become more efficient, more satisfied, and less dependent on other platforms (Piyatissa, 2020).

2 Literature review

Rapid advancements in digital technology have led to innovations that allow companies to utilise the Internet to manage customer orders in an improved manner; leading to significant transformations in the field of food delivery services. Through the ability to place orders using digital means, online order processing systems have proven to be highly instrumental in improving the convenience of consumers (Chavan et al., 2015), as well as operational efficiency, in restaurant management systems (Oghenekaro & Okafor, 2023). This, in turn, aids in the growth of the corporation.

However, despite their numerous benefits, smaller companies face several challenges when using platform-type applications like Wolt and Uber Eats. First, such services can be costly to operate and offer little flexibility or customisation (Piyatissa, 2020). Moreover, businesses have little control over customer information and the app itself. This implies the necessity to developing cheaper and more flexible solutions for smaller firms specialising in food deliveries (Oghenekaro & Okafor, 2023).

2.1 Functional and technical requirements of web-based ordering systems

One of the major aspects of modern e-commerce is the use of online ordering systems, which allow clients to peruse menus, order food, and provide delivery details through an interface (Chavan et al., 2015). The system is composed of two main parts: the backend and frontend. The backend is the part of the system that performs order processing, menu management, and data storage, while the frontend is the client interface (NebiOğlu, 2020).

Order placement, shopping cart support, menu display and customer information management are the primary functional needs of such systems. These features ensure that customers are able to complete their transactions effectively and that organisations can run their operations effectively (Chavan et al., 2015). Since poorly designed interfaces can negatively impact customer experience, usability is a significant factor (Piyatissa, 2020).

From a technical perspective, client-server models are often employed when creating order systems for web platforms. In a client-server model, a server that processes requests and communicates with a database is linked to the frontend of the application. Owing to their low cost and ease of use, PHP and MySQL are often employed when creating small-scale

order systems for businesses. These tools make it easy facilitate information management and the generation of dynamic content (*Mozilla Developer Network*, 2025).

Systems designed for large platforms and those designed for small businesses, however, clearly have differences. This is particularly so in the fact that large platforms have complex features such as payment systems and tracking systems which make the system more complex. On the other hand, small businesses benefit from systems designed to focus on the essential components of the system because they are more affordable and easier to manage (*OECD*, 2023). Therefore, this points to the need to develop systems for small food delivery businesses.

2.2 Rapid application development (RAD) and system architecture

There is a significant relationship between the development process methodology and system efficiency. Classic models, such as the waterfall model, use a linear development method, whereby a step must be concluded before another can be undertaken. The method is systematic but inflexible and does not apply to projects whose requirements evolve as the process does (Senarath, 2021).

These agile methods try to overcome through iterative development, encouraging input along the way. However agile methods often require teamwork and project management which may not be applicable for individual or small scale projects (Senarath, 2021).

Under such a condition, a better approach would be to use Rapid Application Development where software developers can quickly develop and improve system components through rapid prototyping, iterative development and continuous user feedback. In this regard, RAD is particularly useful for web-based systems, where system functionality and usability are constantly being reviewed and improved (*IBM. RAD*, 2024).

Another significant aspect of building effective applications is the system architecture. The system was partitioned into display-, level application-, level and data-level based on the three-tier architecture idea. The maintainability and scalability of the system organisation are improved by partitioning the system into layers. Although more complex architectures, such as micro services, offer better scalability, not all applications require such complexity, particularly at a small scale. Three-tier architecture offers a decent trade-off between usability and simplicity (*IBM. Three-Tier Architecture*, 2021).

Developers can develop systems that are both flexible and well-organised by using a combination of RAD and structured systems. This method ensures the stability of the systems while developing them rapidly.

2.3 Reducing dependency on third party platforms

Third-party food delivery platforms have gained traction in the food service sector because they provide companies with access to a large customer base. However, third-party food delivery platforms have also brought various challenges to the food services sector, especially to small firms (Piyatissa, 2020)

Hefty commission charges associated with these platforms, which can cause a significant reduction in profit margins, are one of the major hurdles. Moreover, the control of the branding and pricing strategies of the business, along with customer data, is limited for businesses that use such platforms (Huang & Khantong, 2024). Therefore, business operations are dependent on such platforms.

Small and medium-sized enterprises benefit from the use of independent digital technologies to improve the efficiency and control of their operations, as suggested by the digital transformation literature. This is because enterprises can operate more efficiently and reduce their dependency on third-party platforms by developing their own web-based applications (OECD, 2023).

Furthermore, autonomous solutions provide more freedom than standardised platforms. For instance businesses can improve the happiness of their customers through the customisation of their systems which will in turn improve the quality of their services. In addition digital tools make small businesses more competitive thus running efficiently (World Bank, 2022)

But it is essential to note that there are advantages such as logistics and a broader market that third party platforms offer. Therefore the aim is to provide an alternative solution that allows small businesses to operate independently rather than replacing the platforms.

2.4 Comparison of food ordering systems

Ordering food using traditional methods is done manually, using methods such as telephone calls for ordering, which are very inefficient and likely to cause mistakes. This makes it highly inefficient because the process does not involve any form of automation or mechanisation. Modern web systems, however, ensure efficient order processing and better customer-business interaction.

A comparative analysis of traditional ordering processes, third-party portals, and the suggested online platform is provided in Table 1 with respect to various considerations. These considerations include order processing, information handling, cost considerations, and system control, which play a major role in determining the performance of the food delivery system. Through such considerations, the table highlights the pros and cons associated with each approach and, thus, underscores the importance of adopting an independent yet cost-effective system for small food delivery enterprises.

Table 1. Comparison of food ordering systems

Features	**Traditional System**	**Third party Platforms**	**Proposed web-based System**	**Justification**
Order method	Phone / manual	Mobile app / web	Web-based	Manual ordering relies on human interaction, leading to inefficiencies, whereas digital systems enable faster and remote ordering (Egereonu, 2024).
Menu display	Printed / verbal	Digital with images	Digital with images	Paper-based menus are static and require reprinting, whereas digital menus are dynamic and can be easily updated (Chavan et al., 2015).
Order accuracy	Low	High	High	Manual systems often result in incorrect or missing orders, whereas automated systems

				reduce human errors (Wang, n.d.).
Order tracking	None	Real time GPS	Status based	Digital systems provide tracking features, although simpler systems use status updates suitable for small-scale operations (Wang, n.d.).
Delivery management	Manual	Integrated systems	Basic management	Traditional systems rely on staff coordination, while digital systems streamline delivery processes (*Hossain, Md Fahad*, n.d.)
Customer data	Not stored	Platform controlled	Business controlled	Digital systems allow the storage and analysis of customer data, improving service quality and decision-making (Piyatissa, 2020)
Cost	Low	High (commission)	Low	Third-party platforms introduce on-going operational costs, whereas independent systems reduce long-term expenses (Oghenekaro & Okafor, 2023)
Control	Full	Limited	Full	Platform-based systems restrict autonomy, while independent systems enable full operational control (Oghenekaro & Okafor, 2023)
Customisation	Limited	Restricted	High	Independent systems allow feature customisation according to business needs

				(Oghenekaro & Okafor, 2023).
Complexity	Very low	High	Moderate	Advanced platforms include complex integrations, whereas smaller systems focus on essential functions (Oghenekaro & Okafor, 2023).

The analysis provided in Table 1 shows, evidence of a change from the traditional manual process to more digitalised procedures. The former approach involved a lot of human contact, thus delaying orders, creating communication problems, and increasing the possibility of placing incorrect orders. According to past research, manual systems face challenges when placed during busy hours, leading to incorrect or incomplete orders (Wang, n.d.).

Third-party platforms,, offer complex functions, including integrated delivery management systems, automated order processing, and live order tracking. By reducing wait times and errors, these capabilities significantly enhance operational efficiency and customer satisfaction. However, these benefits have some significant drawbacks. Third-party platforms impose commission-based pricing methods, limit control over client data, and increase reliance on outside services, all of which can have a detrimental impact on the profitability of small enterprises (Oghenekaro & Okafor, 2023).

In this way, the suggested online platform can overcome these deficiencies by integrating key features of the digital world without increasing complexity or dependency on others. The platform is superior to traditional platforms because it improves the accuracy and availability of orders using a digital format. In addition, the system allows complete ownership of customer data without relying on any outside party (Piyatissa, 2020).

Moreover, the proposed system is based on the principles of simplicity in terms of using status-based tracking and rudimentary delivery management, in contrast to advanced real-time logistics solutions. This strategy supports the findings of studies stating that smaller systems can be effectively designed without excessive complexity but remain functionally efficient (Oghenekaro & Okafor, 2023). The developed solution ensures usability, cost-effectiveness, and operational autonomy.

This implies that the solution presented herein is aimed at serving as an alternative solution and not completely eliminating other third-party solutions. In this way, small businesses can function independently without compromising functionality when dealing with the order process. Such a balance is the solution to the limitations of existing systems have been facing.

2.5 Research gap and contribution

According to the literature, most existing studies focus on broad digital transformation strategies or platforms. There is little emphasis on viable and cost-effective technologies designed particularly for small food delivery companies.

This thesis aims to fill this void by creating a streamlined web-based ordering and delivery management system that emphasises key features. This system has been designed to be cost-effective, user-friendly, and suitable for small businesses with tight budgets.

This thesis contributes to the literature by demonstrating how a simple solution can be applied to meet real business needs without unnecessary complexity. The proposed solution provides an important method for creating autonomous digital solutions for small food delivery companies by combining the principles of Rapid Application Development and system design.

3 Digital technologies for web-based ordering system

A range of digital technologies are employed in web-based ordering and delivery management systems to enhance user system interaction. These technologies make it easier to place orders, manage data, browse menus, and run the system effectively. The old-school approach that involves using telephone calls for placing orders is not only inefficient but also error-prone; and therefore, such systems have increasingly been replaced by digital systems in the food delivery business. Sophisticated digital systems with useful capabilities have been offered by massive organisations such as Wolt and Uber Eats. Nevertheless, these systems may not be suitable for small businesses involved in food delivery because they are usually costly and complex. This system is designed to use simple and affordable digital technologies (*OECD*, 2023).

3.1 Structure of web-based ordering system

There are three main components of a web-based ordering system: the interface, backend, and database. While the backend component processes requests from users and manages operations within the system, the frontend acts as the interface through which clients can access the system. Data relevant to the operations of the system, which include order information menu items and customer information, among others, are contained in the database. Collectively these components ensure that the system operates efficiently. This efficiency is achieved when the backend works the request while the frontend captures the input from the user (*Mozilla Developer Network*, 2025).

3.2 Frontend technologies

The frontend will be tasked with creating the GUI and will be very important in determining the user interface of the application. The technologies HTML, CSS, and JavaScript have been used in the frontend development of the present project to make the interface user-friendly. Users will be able to efficiently navigate through the menu select the products input delivery details and place orders using these tools. Good frontend design helps in creating a usable interface which is important for a satisfying user experience (*Mozilla Developer Network*, 2025)

HTML is used to define the structure of the system through the organisation of menu bars, forms, buttons and input fields, Through its ability to enable customers to see the items on offer and enter their order information it acts as the foundation for customer interaction. With the help of CSS, the appearance of the system can be enhanced by controlling aspects such as layouts, colours, fonts and spacing. With the help of JavaScript there can be interaction which allows for an immediate response to any action made by the person using the system. For example, the ability to validate data entry from the client side and add items to the cart can be done immediately without reloading the web page.

3.2.1 Usability and system design principles

The ease of use of a website is crucial for the proper functioning of a web-based system, especially in ordering systems, where ease of use effects how users interact with the system. According to the theory of usability, an ideal system is easy to learn, efficient, and offers a satisfactory user experience, which reduces user frustration and mistakes (Nielsen, 1994)

According to TAM (Technology Acceptance Model), it is important to note that users' acceptance of technology is dependent on two variables which include perceived ease of use and usefulness of technology (Fred D., n.d.). In terms of an online ordering system, it means that the adoption of this technology will be highly dependent on whether users find this technology useful and easy to use.

Usability guidelines were employed in this research to design an appropriate navigation system, easy-to-use order process, and user-friendly interface. Important aspects like categorisation of menu items, easy access to the shopping cart, and timely system feedback have been incorporated to ensure that users find navigating through the system easy and efficient (Jesse James, 2011)

3.3 Backend technology PHP and server side processing

The basic operations that include dealing with user queries and communicating with the database are carried out using the backend technologies. In this case, PHP is the server side scripting language that is used extensively. When the user interacts with the system, say through placing an order the data collected from the frontend is passed to the server and is processed there through PHP.

Through processing tasks such as recording order records, obtaining menu records and processing client records PHP assumes a vital role in directing the logic of the system. Furthermore, it allows dynamic content generation, allowing the system to display dynamic data based on user activity. The fact that PHP can work together with web servers such as Apache and database systems such as MySQL making it suitable for developing websites is an essential advantage of PHP (*Mozilla Developer Network*, 2025). The selection of PHP in this system was driven by its lightweight nature, simplicity and capacity to provide all the necessary features for a small-scale order management system.

3.4 Database management

As it is responsible for managing all data related to the applications it is very important that the database is included in the system. There were several reasons why the MySQL database management system was chosen for this project, but foremost among them was the consistency and simplicity of the MySQL.

All information regarding orders, menus, and customers will be stored within the database to ensure that the data are organised. Efficient management relies on the capability of the system to maintain data integrity and accuracy using a relational database design. The connection between MySQL and PHP allows seamless exchange of data between the database and the back end (*IBM. (n.d.). Three-Tier Architecture*, 2021).

3.5 Development environment XAMPP

The XAMPP (Cross-Platform + Apache + MySQL + PHP + Perl) environment will be employed as the environment within which the system will be developed. The Apache web server, MySQL and PHP capabilities are some of the services offered by the XAMPP environment. This avoids the need for a third-party hosting service and helps create the system locally.

By integrating of the essential software within one programme, the process of developing the application becomes convenient. Using this environment, it is possible to test, troubleshoot and execute the system during its development. Therefore, this environment is particularly useful for developing small systems.

3.6 Integration of system components

These frontend, backend, and database technologies must be integrated for the online ordering application to work. User input was collected and sent to the backend via frontend technology. Communication with the database is performed during request processing to either store or obtain data. Subsequently, the frontend displays the results of this interaction.

The effective functioning of the application is secured through continuous communication between different elements of the system. Data processing, real-time communication and good performance are enabled by the use of such tools (*Mozilla Developer Network*, 2025).

All the digital technologies used to develop the web-based order management system are discussed in this chapter. All the required technologies to make a perfect and efficient system include the frontend technology, PHP backend technology, MySQL database technology and XAMPP development tool. Such digital technologies were selected to ensure that the developed system remains affordable, convenient to use, and applicable to small food delivery firms. The system development process is explained in detail in the next chapter.

4 System design and implementation

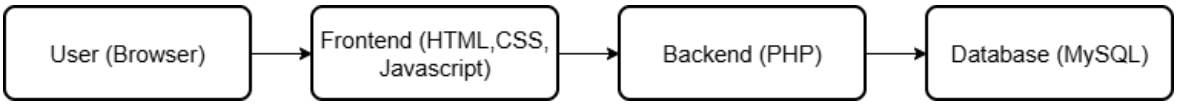
This chapter focuses on the design and implementation of the online food ordering and delivery management system that was designed in this dissertation. This framework provides an easy way for small businesses to manage their customers' orders through the use of a web-based application where customers can check menus and place their orders while receiving information about the status of their orders. The system comprises frontend, backend, and database layers.

The process will be designed according to usability guidelines and the RAD approach in software design, where iterations can be applied throughout the entire process, resulting in improvements at any step of system development. In addition, the design will incorporate only the necessary functionalities.

4.1 System architecture

The system has three-level architecture, shown in the figure below labelled Figure 1. System architecture. There are three levels: display level (frontend), application level (backend), and data level (database).

Figure 1. System architecture

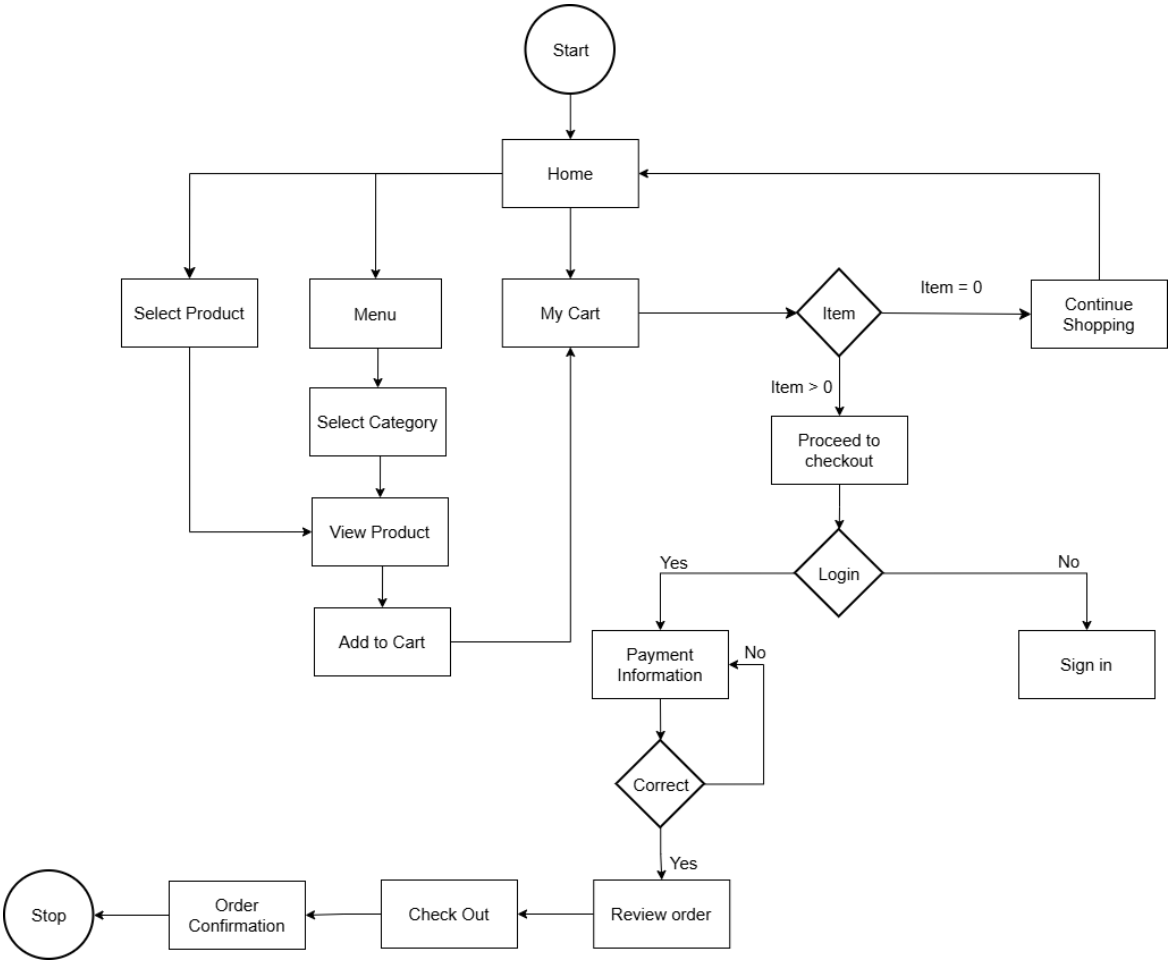


The user interface, including the menu presentation and user input, is handled by the front end. The request is processed by the backend, which also transfers instructions to the database layer. The system data are stored in the database layer.

4.2 System workflow (User flow)

The total process can be carried out in a systematic manner, where the user goes through all the stages from browsing to purchasing orders.

Figure 2. System activity workflow



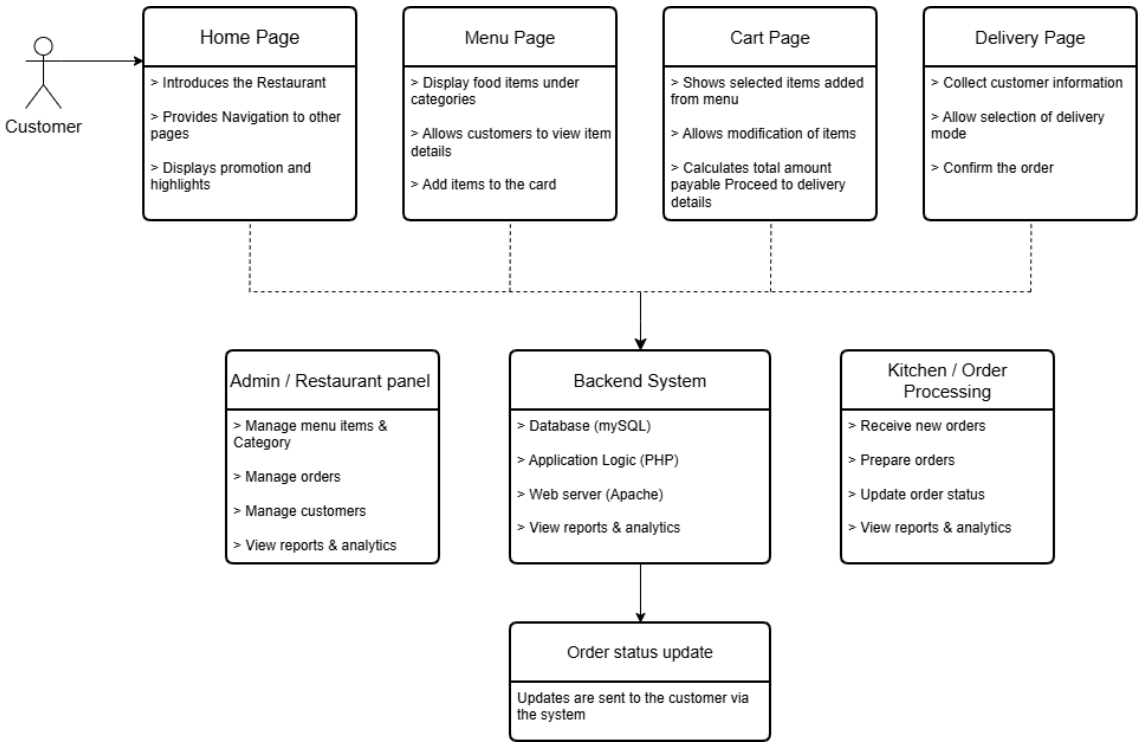
The total process can be carried out in a systematic manner, where the user goes through all the stages from browsing to purchasing orders.

Figure 2 shows how the workflow is set up for users to order products conveniently. There are specific steps included in the process for easy usability, while the use of local storage helps in retaining cart items.

4.3 System components and functionality

The Figure 3 shows the various parts involved in the ordering process. The home page component introduces the customer to the restaurant and directs them to other webpages. The menu page component has food items listed under their categories; here items can be added to the cart. The cart page component is where the customer views the items selected on the previous page modified the number and checks the total amount payable. In the delivery component, the user inputs details like name, contact number, and address while choosing the mode of delivery. All the above part form a working system that ensures a good user experience during ordering.

Figure 3. System components



4.4 Database design (ER diagram)

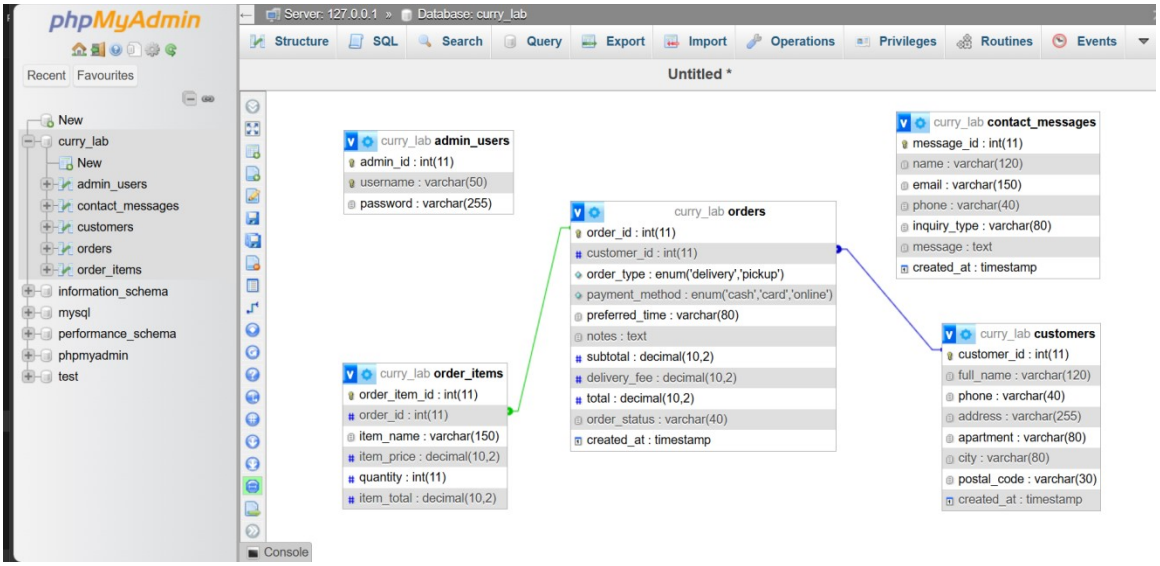
Relational database design is extremely important for the efficient storage of data, consistency, and scalability of the system. In this project developed a relational database in MySQL, which is responsible for storing data about customers, orders, items,

administration, and contact forms. The database is built using the ER model, which represents the database structure by creating entities and their relationships.

The major entities found within the system include the customer, order, order items, administrator, and contact. The individual entity reflects an aspect of the system that requires a different table to minimise duplication and ensure data integrity. Data normalisation rules ensure that data are separated into several related tables, rather than being placed in one table for all data.

The entity relationship diagram of the system is presented in Figure 4. The ER diagram shows the relationship between various entities and how data are exchanged among them. The ER diagram also depicts the use of primary keys and foreign keys to form relationships between tables.

Figure 4. Entity Relationship diagram



The model is based on a logical structure that includes the following one-to-many relationships:

- One customer may have several orders each order is assigned only to one customer. As such, one-to-many relationship happens between the customer and order entities.
- Each order may include several products, stored in order_items. Here is another one-to-many relationship that goes from orders to order_item table.

- Order items relate to a particular order and denote product choices along with their quantities and prices.
- Contact messages relate to user queries left using the contact page and are not associated with any order in particular.
- Finally, the admin_users table is an independent entity that allows to authentication of users and performs various system management tasks.

The database design was designed to ensure scalability and sustainability. The customer, order, and item databases were divided into various tables to ensure that there was no data duplication. This means that customer data have only been saved once in the customer database, and a foreign key is used in the orders database.

Moreover, the order_items table enables the creation of many items in an order without having to duplicate order-specific attributes such as the type of delivery and mode of payment. This provides flexibility in the management of complex orders.

The relational model also ensures data integrity aspect as well. The foreign key controls guarantees that each order is related to an existing customer and each order line is related to an existing order.

Overall, the database design serves as a systematic framework to support the entire process. The application of the relational model helps ensure that information is systematically arranged and not repetitive. It also creates room for expansion, whereby new features can easily be added to the database in the future without redesigning.

4.5 System implementation

This technology is achieved by utilising various Web programming languages such as HTML, CSS, and JavaScript on the front end, PHP on the back end, and MySQL on the database side. These components were used in the implementation of this project in the XAMPP environment.

User interaction is performed by the front-end which sends requests to the back-end which processes care of processing those requests, communicates with the database, and performs data storage or retrieval.

4.6 Order processing logic

The order processing algorithm is systematic, for instance when a consumer adds items to their shopping cart, the data are saved locally before being retrieved and shown on the delivery page. After the form is submitted, the system generates an order after confirming the data are correct.

The system will creates a receipt and simulates order tracking steps such “Order Received,” “Order Preparing,” and “Order Delivered” to enhance the user experience.

This chapter describes the design and implementation of the online ordering system. The structure, process, and design of the database are explained to understand how the system works. The effective operation of the system involves the use of a combination of the front-end, back-end, and database together. The following chapter provides an evaluation of the system and its results.

5 System implementation and testing

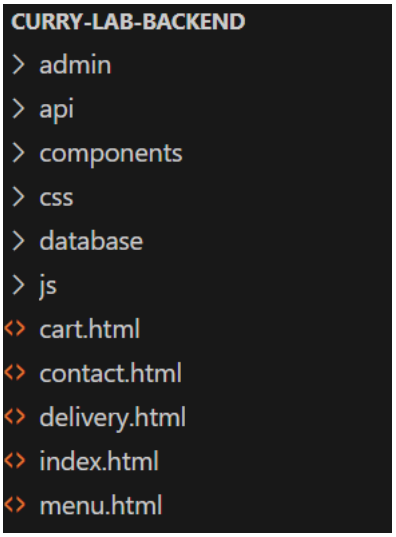
The implementation of the food ordering system developed for the Curry Lab is discussed in this chapter. The aim is to discuss how the system is developed and how its components work to support the research objective of developing a standalone online ordering platform for small businesses. It also covers front-end design, back-end processing, database connectivity, order processing, and contact form processing

HTML, CSS, JavaScript, PHP, and MySQL were used for developing the system. The purpose of HTML was to define the structure on the web page. CSS was used for designing the look and feel of the system. JavaScript was used for handling interactions between the user and cart actions. PHP was used for handling server-side requests, while MySQL was used for storing the data related to orders and customers. The system contains home page, menu page, cart page, deliver/checkout page, contact page, and admin order management page.

5.1 Project structure and file organisation

The program was organised using a modular design approach. Rather than building it as a single-file software entity, the project was developed in several modules, such as front-end pages, UI components, styles, JavaScript code, back-end scripts, and database schema.

Figure 5. Overall project folder structure



This architectural choice was deliberate. In smaller-scale projects, everything is usually in one or two files; however, this practice results in tightly integrated code that is not easy to troubleshoot or expand. This project uses separate components for each concern, making it more realistic in relation to actual web development, where the front end, logic, and back end are managed separately as illustrated in the above Figure 5.

For instance, the division of CSS into base.css, layout.css, and components.css enables each of them to have its own functionality. The base will be concerned with the overall style of the website including colours and font type while the layout handles the positioning of content and grids, and finally, the components take care of all the reusable interface components like buttons and cards.

Similarly, the implementation of a dedicated folder named api for back-end processes and another folder called admin for administrative purposes indicates the proper structuring of the application. The benefit of this approach is that it makes the code much clearer, while in the future, other capabilities, such as authentication and analytics, may be easily added to the project.

Table 2. Overview of system components and its functions

Section	**Files /Folders**	**Purpose**
Frontend pages	Index.html, menu.html, cart.html, delivery.html, contact.html	Main user interface pages
Shared CSS	Base.css, layout.css, components.css	Global styling, layout, buttons, cards, forms
JavaScript	Common.js, menu.js, cart.js, delivery.js, contact.js	Interaction, cart handling, form logic
Components	Header.html, Footer.html	Reusable navigation and footer
Backend	Api/	PHP processing and database connection
Admin	Admin/	Admin order viewing and login

Table 2. Overview of system components and its functions shows the organisational structure of the proposed system, indicating that there is clear demarcation of the frontend, backend, and associated components as well as their responsibilities. The system has

been designed with the help of a modular design, which involves dividing the different parts into separate folders and files. The benefit of such an arrangement lies in making changes easier because any changes made to one file will not affect other parts. The role of the frontend pages involves interacting with users and displaying content on the screen, whereas CSS files handle the visual design.

Moreover, JavaScript files provide dynamic functionality including cart, user interactions, and form validation. It can be achieved through common JavaScript files wherein the functions that can be reused on other pages are used once in a page. In addition, the reusability feature is further strengthened by the reusable elements like the header and footer.

PHP scripts process data and interact with the database in a different folder named "api," which has backend logic. In this manner, the database management and backend logic of the application are kept apart from the user interface. Additionally, there is an admin folder with order management and user authentication features.

5.2 Reusable header and footer implementation

Header and Footer were added as reusable components to ensure consistency across the website. Figure 6 consists of Curry Labs' logo, tagline, and links to the Home page, Menu page, Shopping Cart, Delivery page, and Contact page. Each menu item has a 'data-page' attribute which will help JavaScript recognise the active page.

Figure 6. Header and navigation bar

Figure 7 includes the copyright details and brand statement of the restaurant. The footer as a standalone entity will ensures that it can be reused on all pages without duplicating the code.

Figure 7. Footer section

This reusable component system adds flexibility. For instance, if there was a need to make any modifications to the navigation bar, the developer would only have to make such modifications on the header.html file and not every single file separately.

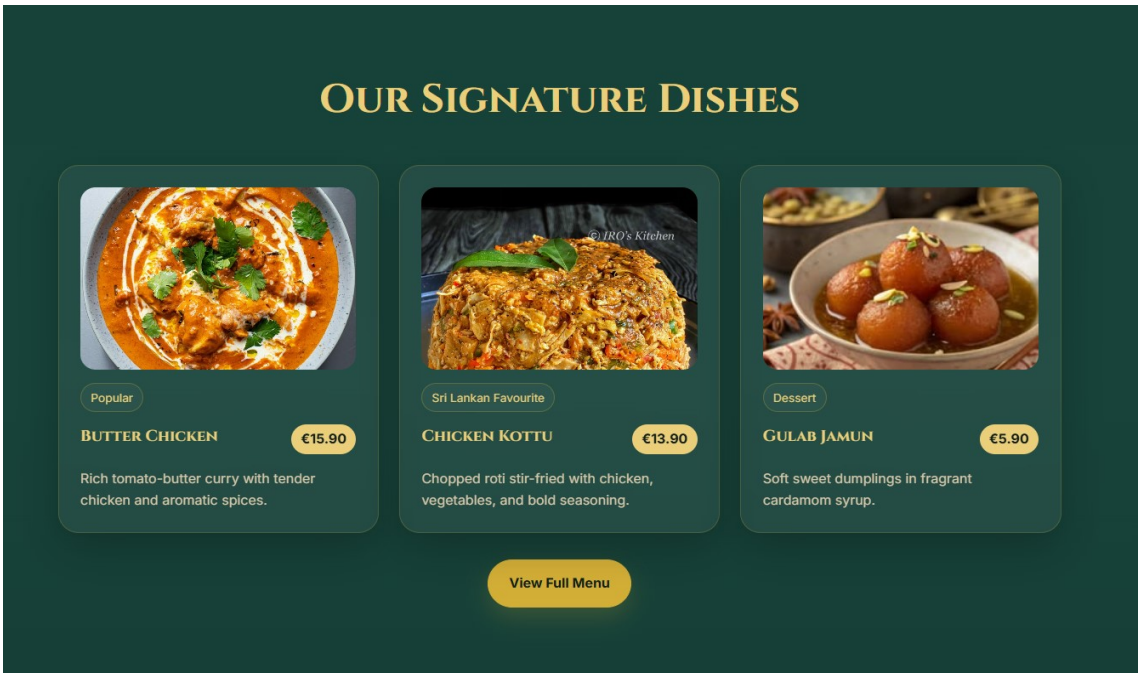
5.3 Home page

The home page is the initial point of contact for users interacting with the Curry Lab ordering system. The restaurant is introduced using a hero section, brand message, featured items, benefits of services, process, review, and call-to-action buttons. It is crucial to understand that the design of the home page plays an essential role in creating the first impression of the ordering system.

Figure 8. Home page hero section

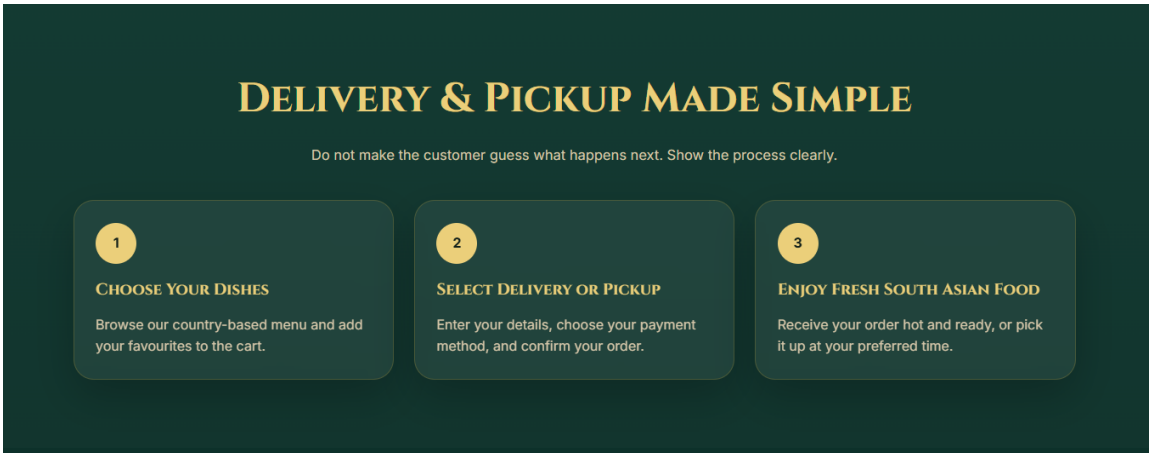
Figure 8, explains the use of hero design on the home page which involves the inclusion of a heading, supporting details, and call-to-action buttons directing people to other sections such as the menu and contact pages. Restaurant highlights featured on the home page include ratings, average delivery time, and freshness, among others. They are important for conveying trust and service provision information. Figure 9 indicate the featured items, including Butter Chicken, Chicken Kottu, and Gulab Jamun allows visitors to know about the type of foods served.

Figure 9. Featured dish section



The home page also contains another important section shown in Figure 10, entitled “Delivery & Pickup Made Simple”, which describes the process in three steps: selection of dishes, choice between delivery or pickup service, and enjoying the meal. This feature makes the website more usable, because it decreases ambiguity for users and provides information prior to placing an order.

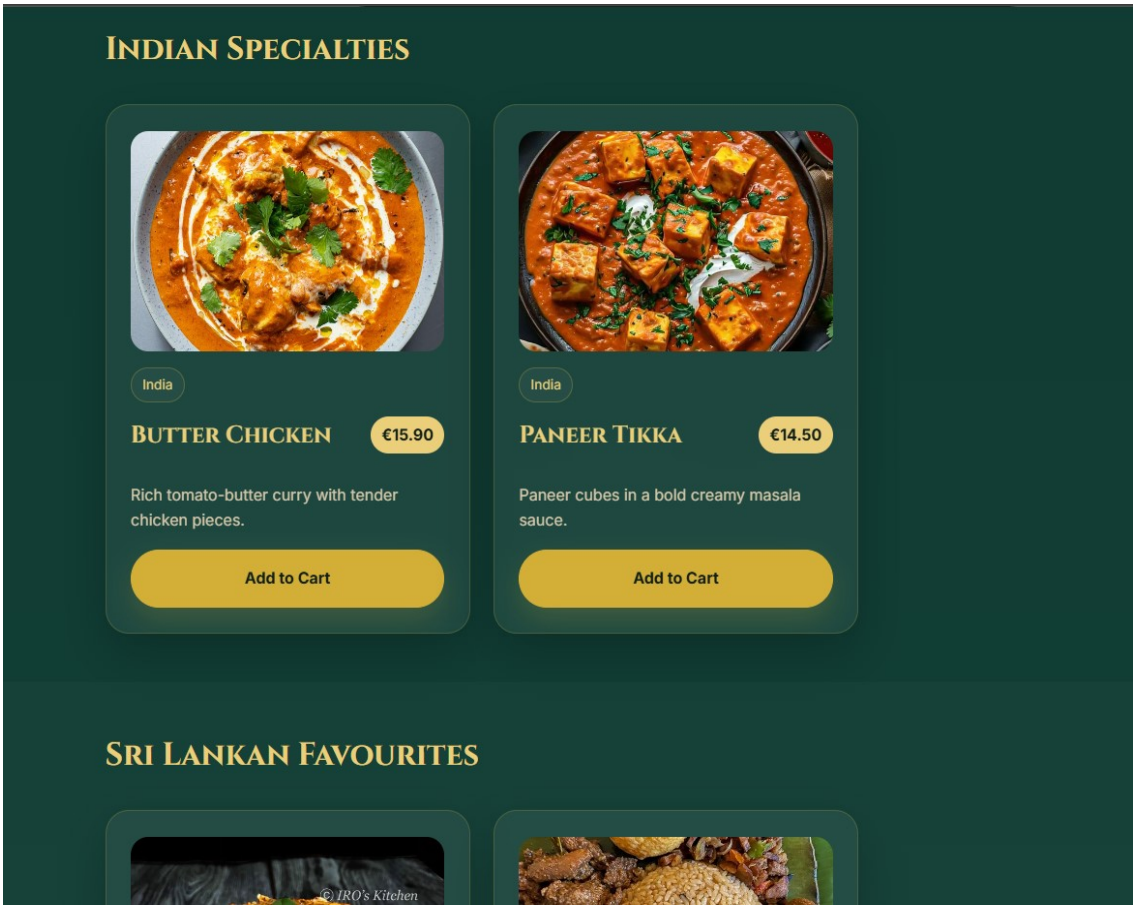
Figure 10. Delivery and pickup process section



5.4 Menu page

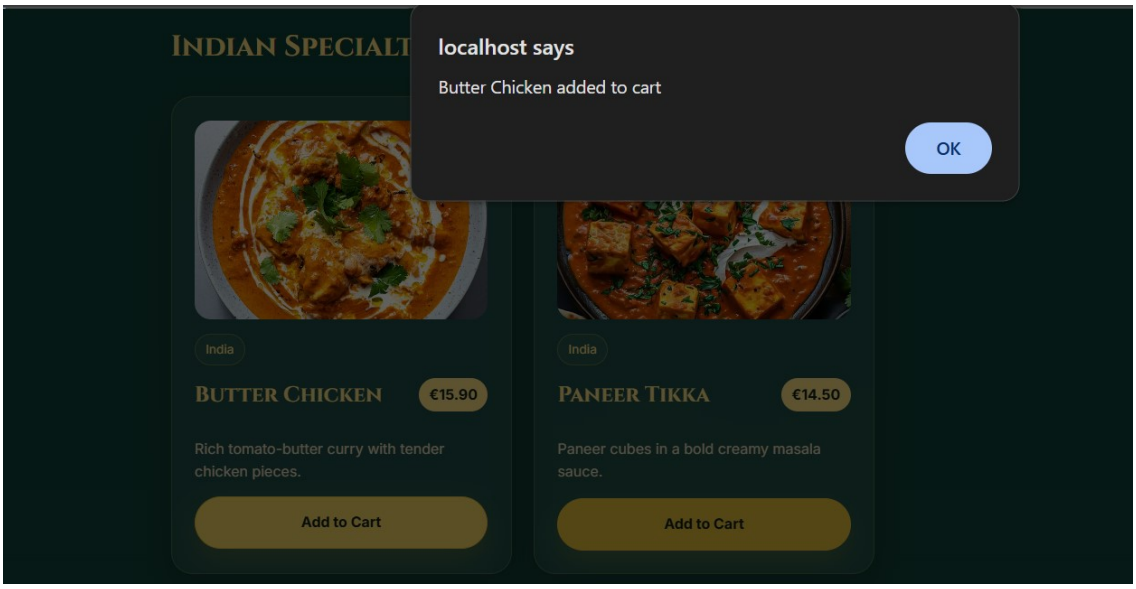
Figure 11 describes how the menu page is implemented and exhibits how client-side interaction matches real-time e-commerce without requiring back-end input from the beginning. Metadata for each menu item is added to HTML code using custom attributes to enable processing by JavaScript.

Figure 11. Menu page with item cards



This makes it unnecessary to write hardcoded code for each item. Rather, a common event listener can handle all "Add to Cart" events by simply extracting information from the element that is clicked as shown in Figure 12. The system is scalable because more menu items can be easily added later.

Figure 12. Add to cart button interaction



The adoption of browser local storage for the management of the shopping cart is the result of a compromise between convenience and dependability. When using browser local storage, the data will continue to remain until the user navigates from one page to another, which makes the browsing process convenient. This is limited to the browser of a specific user and cannot handle multiple browsers.

Another significant aspect of the implementation process is the issue of duplication. Rather than having a number of duplicates, the programme will simply add up to the number of a certain entry in the cart. This is logical from the perspective of e-commerce behaviour, where it is common practice not to have duplicates.

5.5 Cart page

Figure 13 refers to, where customers can see all the selected foods before moving to the checkout process. This page accesses the data of the cart stored in the local storage of the browser and shows all the selected foods together with their prices and quantities along with totals per each selected food.

Figure 13. Cart page

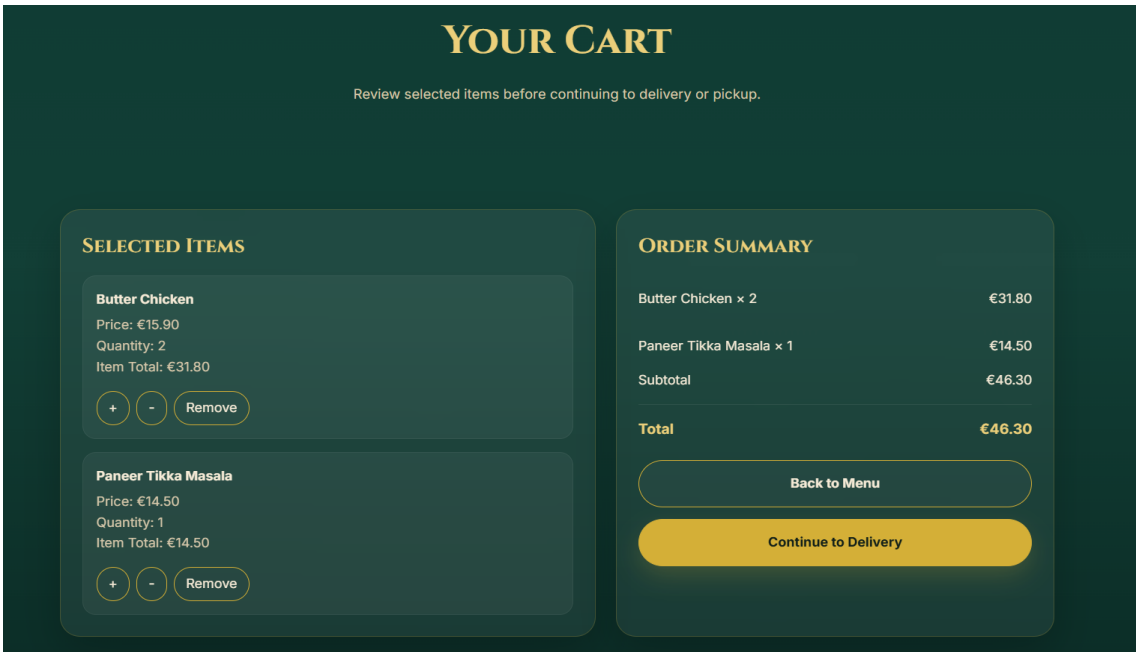
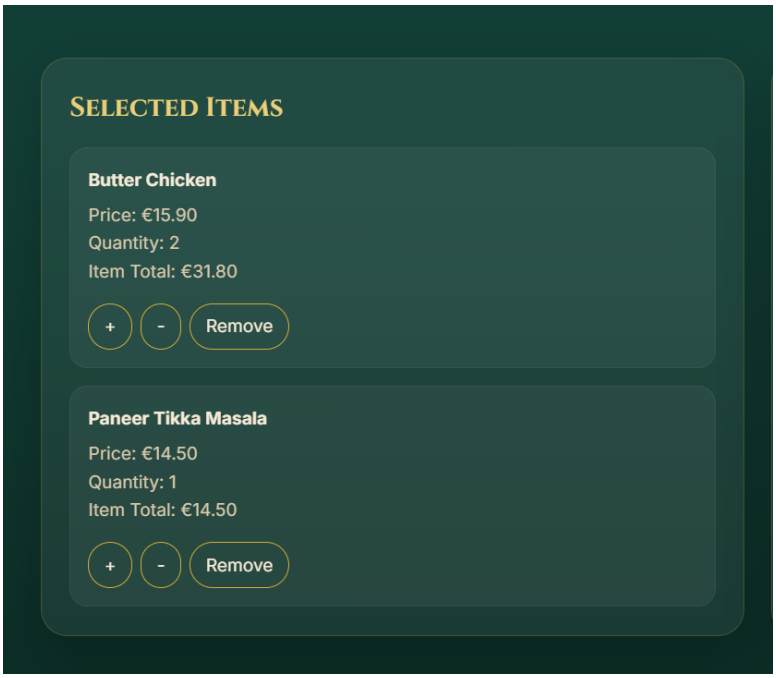


Figure 14 illustrates that add, reduce, and delete functions ensure that the shopping cart reacts dynamically. In the event of any change in quantities, the cart is stored locally and reloads itself. This confirms real-time updating of the cart without reloading the page.

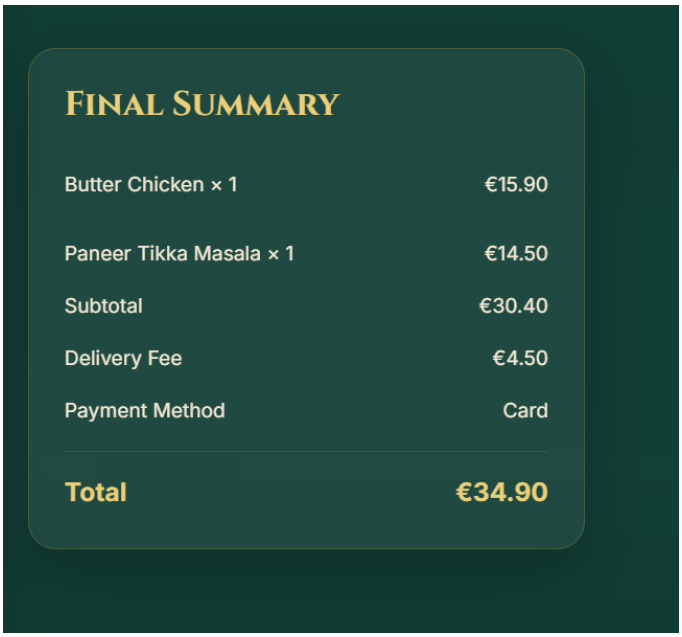
Figure 14. Cart - selected Items



The cart page consists of two parts: selected items and an order summary. Selected items refer to the items in the cart, whereas the order summary in Figure 15 refers to the subtotal and total amount of the purchase. Navigation buttons are present on the page to direct users to either the menu or the delivery page.

The functionality of the cart is coded in the cart.js file. The loadCart() method is used to obtain cart data, remove the current data from the page, calculate the subtotal value, and create cart items dynamically. If the cart contains no products, a message is shows to inform the users.

Figure 15. Cart summary



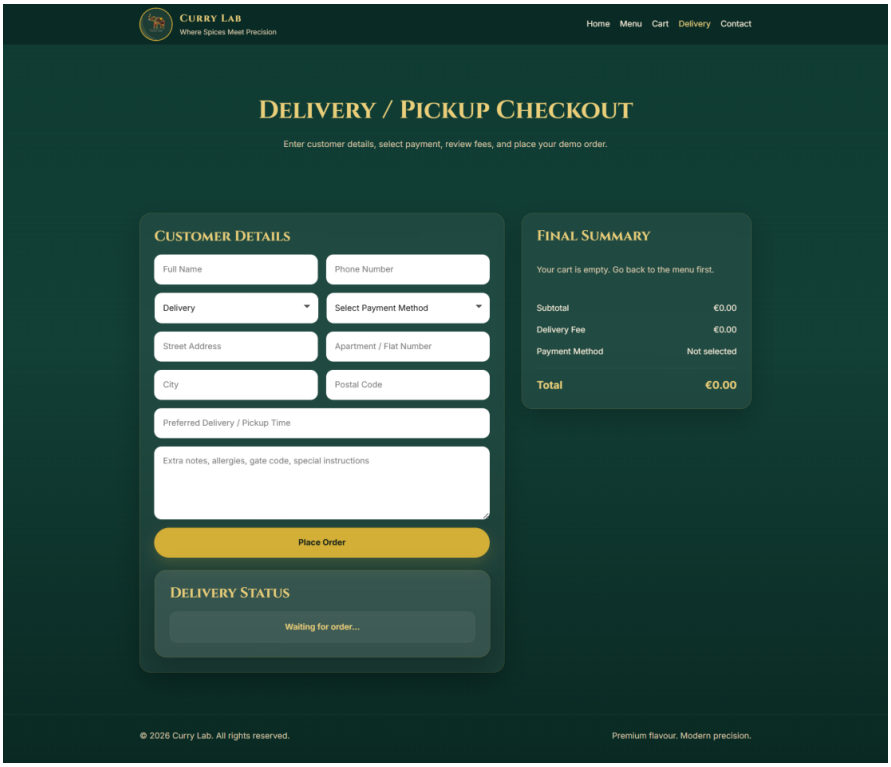
5.6 Delivery and checkout page

Figure 16 is considered to be the most complex interaction level page, because it incorporates all the processes related to user input handling, interface dynamics, validation, and pricing. During the development of this page, certain considerations were taken into account, including providing relevant fields according to the user’s selection.

The first important implementation aspect is related to the dynamic display of input fields depending on the user's selection. If the user opts for pickup, all fields related to the

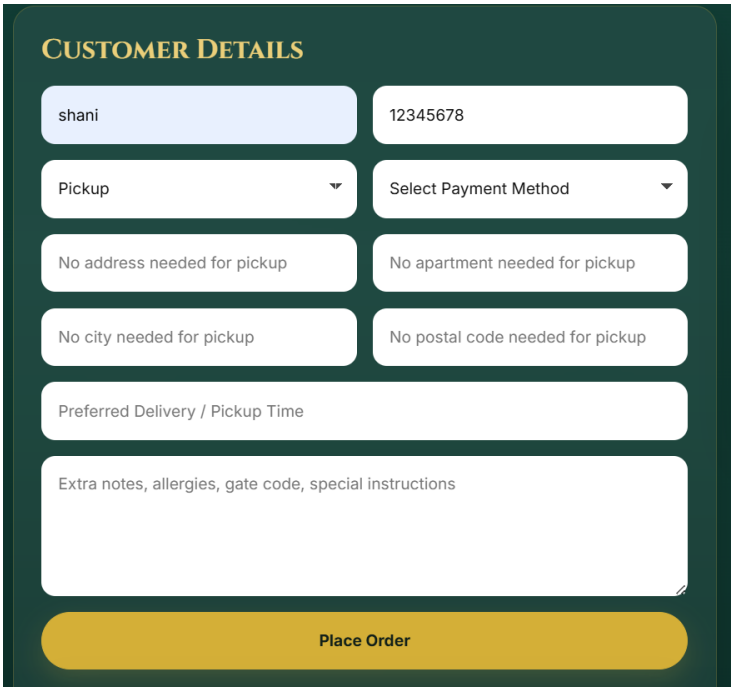
address become unavailable and are erased. In this way, unnecessary actions are prevented and users do not feel confused. However, if the delivery choice is made, such fields must be filled out.

Figure 16. Delivery and checkout page



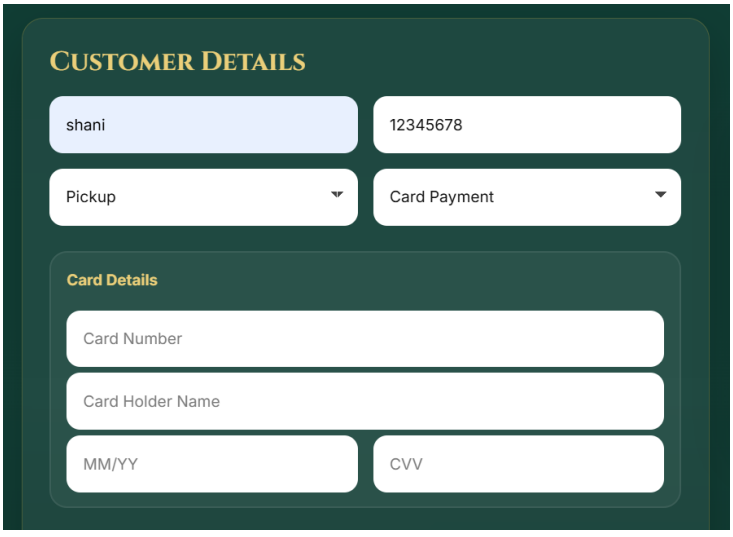
One notable aspect of the design of the delivery and checkout pages is the use of the pickup option, which automatically changes the form according to the user input. When the pickup option is chosen by the consumer as depicted in Figure 17, all the fields that collect the address details of the user, including the street name, apartment number, town, and zip code, are automatically emptied and replaced with messages informing the user that there is no need to enter any information currently. In this way, the consumer does not have to input irrelevant information, thus improving efficiency. Dynamic behaviour was achieved using the JavaScript programming language.

Figure 17. Order Pickup form



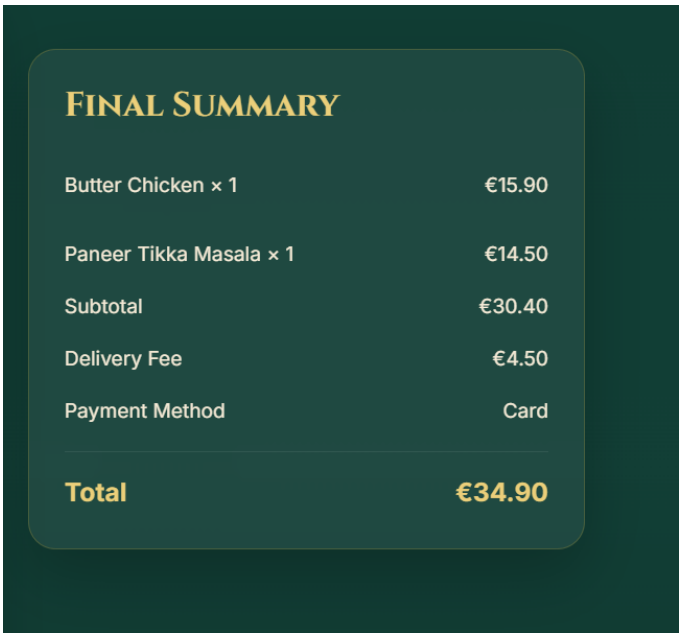
Similarly, the choice of payment method affects the user interface in a conditional manner. If the user chooses to make a payment through a credit card as shown in Figure 18, there will be fields available for credit card information. Using such an approach, the user interface does not become crowded with extra options that are not required.

Figure 18. Card payment form



Moreover, there is an introduction of yet another rule that will be more realistic; this includes adding the delivery charge only if there is an option for delivery. Again, the above process will be performed such that the code is reusable. Finally, the real-time updating of the subtotal, delivery charge, and total shown in Figure 19 ensures immediate feedback on behalf of the user.

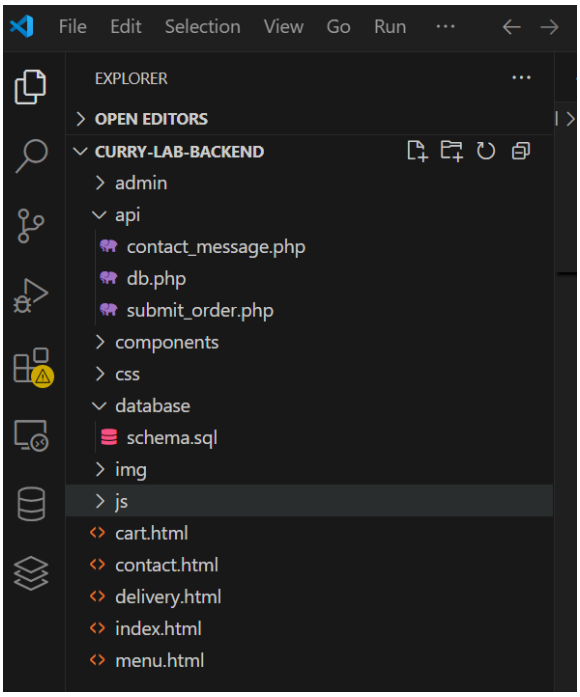
Figure 19. Final summary panel



5.7 Backend order processing

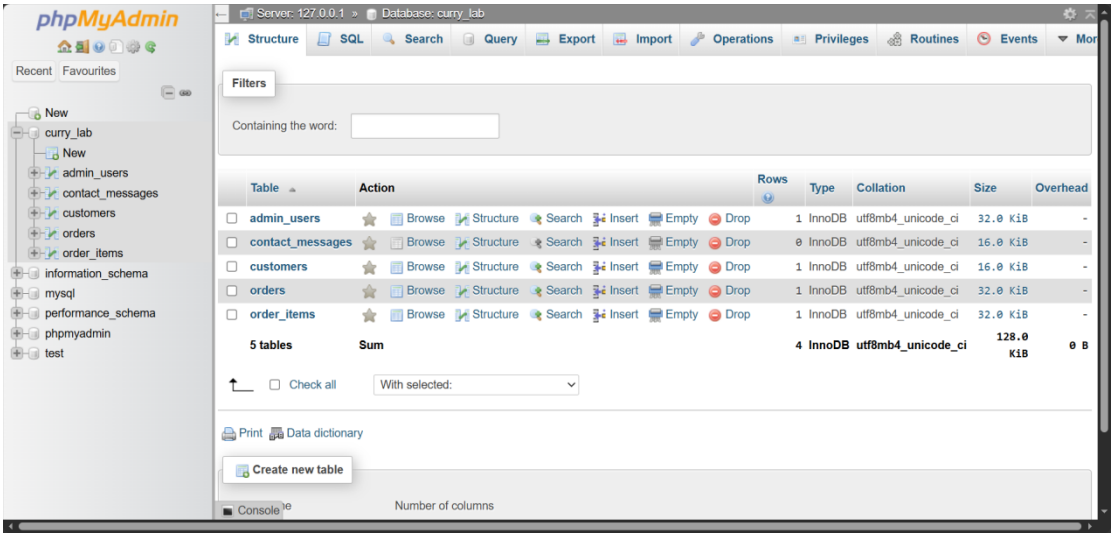
A major increase in system capabilities is represented by the switch from completely client-side architecture to one that is supported by the backend. Although the frontend design effectively mimics ordering behaviour, it is not persistent enough to support actual business processes. These restrictions are addressed by the implementation of a PHP and MySQL backend, Figure 20 illustrates the permitted structured data administration and persistent data storage.

Figure 20. Backend API folder



The backend can deal with structured order information such as customer details, order metadata, and each order item. It does not store all the information in one table; rather, Figure 21 shows that the application uses a relational database system that separates data into different tables for customers, orders, and order items.

Figure 21. Database tables in phpMyAdmin



Upon submission of orders by users, the front end sends all necessary data in JSON format. On the backend, processing takes place through different steps including validation, adding customer details, adding an order, and finally adding items one at a time. This ensures a logical connection between various datasets in the database.

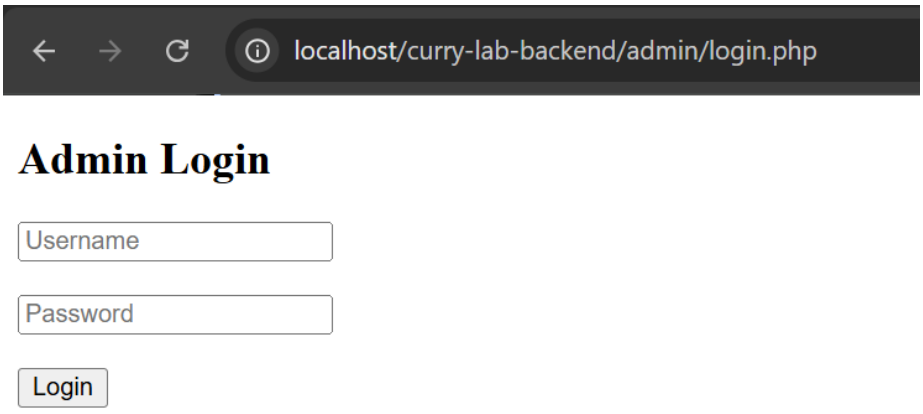
Transactions help ensure that the database receives all entries successfully, and if one entry fails, everything else is rolled back. This shows an appreciation of the concept of reliability and data consistency, which are two dynamic aspects of backend system development.

5.8 Admin login and order management

An admin section was implemented to allow authorised users to view customer orders. This feature is essential for a functional ordering system; as it enables the restaurant to monitor incoming orders.

Figure 22 shows that the login system was implemented to restrict access to the admin page. This verifies that sensitive client data are accessible only by authorised users. Session-based authentication is used to manage the user access.

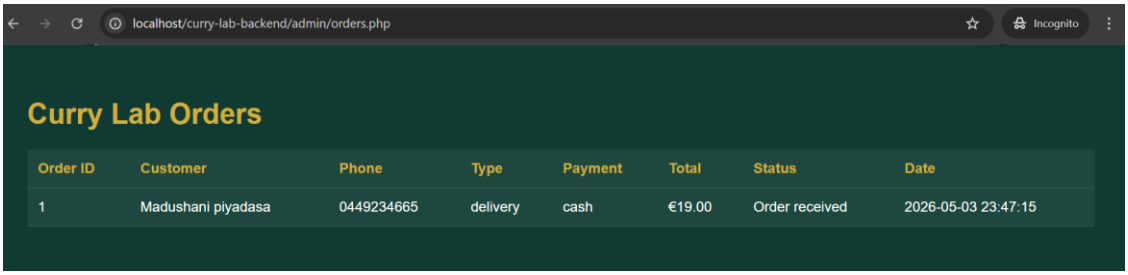
Figure 22. Admin login page



On successful login, the administrator can check all the orders that have been made in a tabulated form as shown in

Figure 23. This will consist of the client's information, type of order placed, mode of payment used, and amount charged.

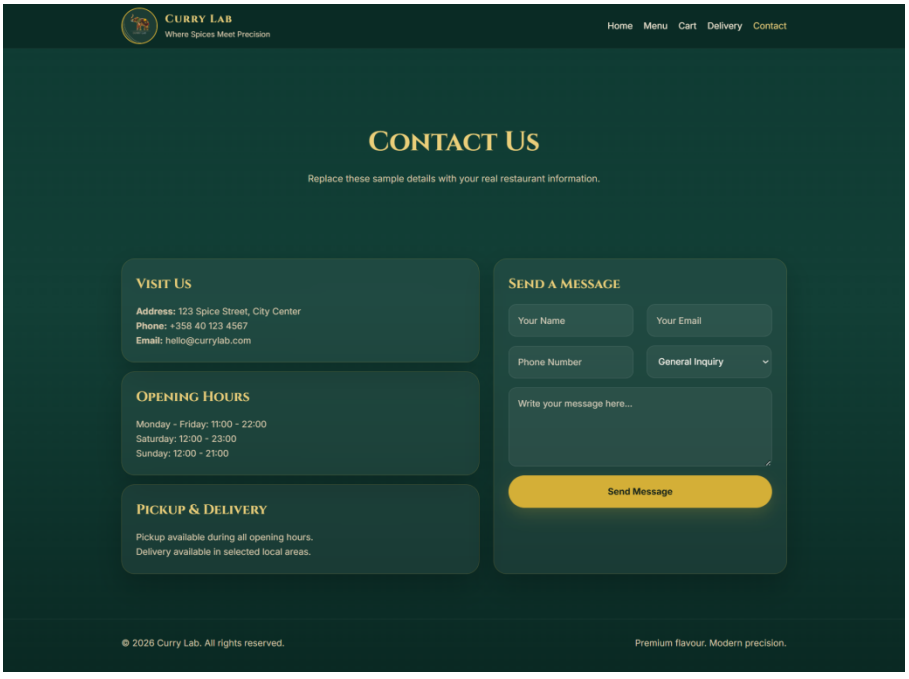
Figure 23. Admin orders page



5.9 Contact page

Figure 24 illustrates information about the restaurant and offers a way for users to communicate with the business. The page also provides information on the location of the restaurant, including the phone number, email, and hours of operation, and pickup or delivery details.

Figure 24. Contact page



The contact form consists of username, email, phone number, type of query, and message boxes. Currently, when the submit button on the contact form is clicked, a demo alert box

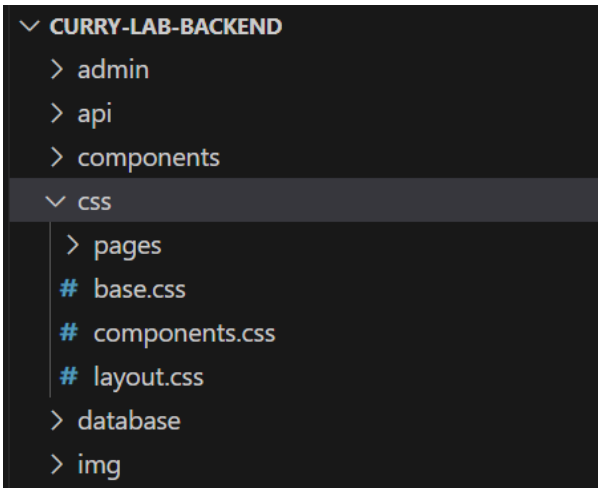
pops up. Contact.js contains the code to detect when a user submits the form, prevent its default action, and display a message to add backend functionality in the future.

If the backend contact data storage option is selected, the contact page form sends the message to the PHP file and stores it in the contact_messages database table. in this way, restaurant employees can see customer requests on the admin panel.

5.10 User interface and CSS

The UI of the Curry Lab website was built based on of the CSS separation model illustrated in Figure 25. The system uses many files instead of combining all styling properties into one style sheet file. Because global styling rules can be reused throughout the system, this method makes modular coding easier.

Figure 25. CSS file separation



The base.css file contains all the basic styles of the system. The colour variables such as the background colour, which is dark green, the highlight colour, which is gold, the text colour, which is cream, card background, border colour, shadow and border-radius, are defined here. Other features include global reset styles, body fonts, how images behave, styles applied to links, form elements, inheritance of form elements, placeholders, and text area behaviour. Therefore, this is the building block of the entire application in terms of the visuals.

The layout.css file manages the layout of the entire website. It is used to arrange the site's structural elements such as the main container width, space between sections, space between page banners, grid arrangement, form rows, and responsiveness of the site. Some of the main areas where layout.css arranges elements include the menu grid, cart layout, delivery layout, contact layout, hero statistics, process grid, and review grid. The media queries allow the columns to become a one-column layout in the case of small screen sizes.

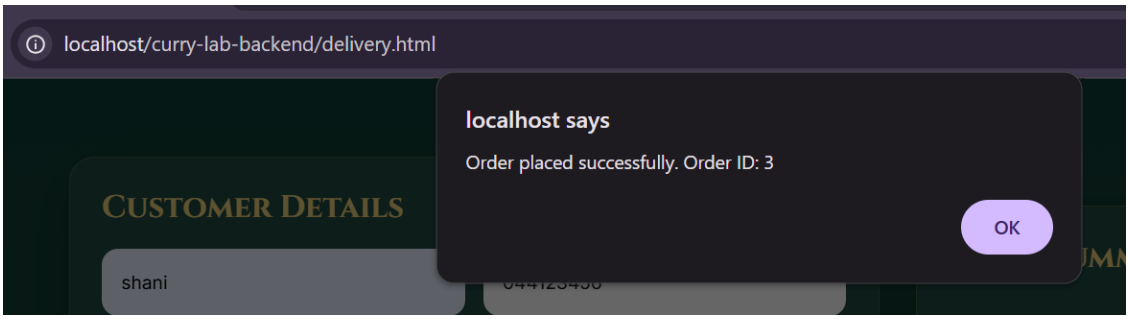
The components found in components.css include interface components used on more than one page. Examples include cards, buttons, navigation bar, summary rows, input fields, empty messages, and footer. This segregation is vital because elements such as buttons, cards, navigation links, and form inputs are repeated within the application. This means that keeping them within a single file helps avoid repetition and ensures that they are styled uniformly across all pages.

Overall, the CSS architecture facilitates separation of concerns because styling duties have been segregated into global, layout, and component styling. It becomes easy to maintain because any change in generic branding can be made through base.css, any alteration in the layout can be made using layout.css, and any reusable interface styling can be altered using components.css.

5.11 Testing of the system

To test the system, a procedure for the complete ordering process was performed starting from the menu page to the admin order viewing page. First, the user accesses the menu page, where the customer selects food items and ensures that these food items have been placed inside the cart successfully as shown in Figure 26. The cart page is then accessed again to check whether the selected items, quantity, subtotal, and total have been displayed accurately.

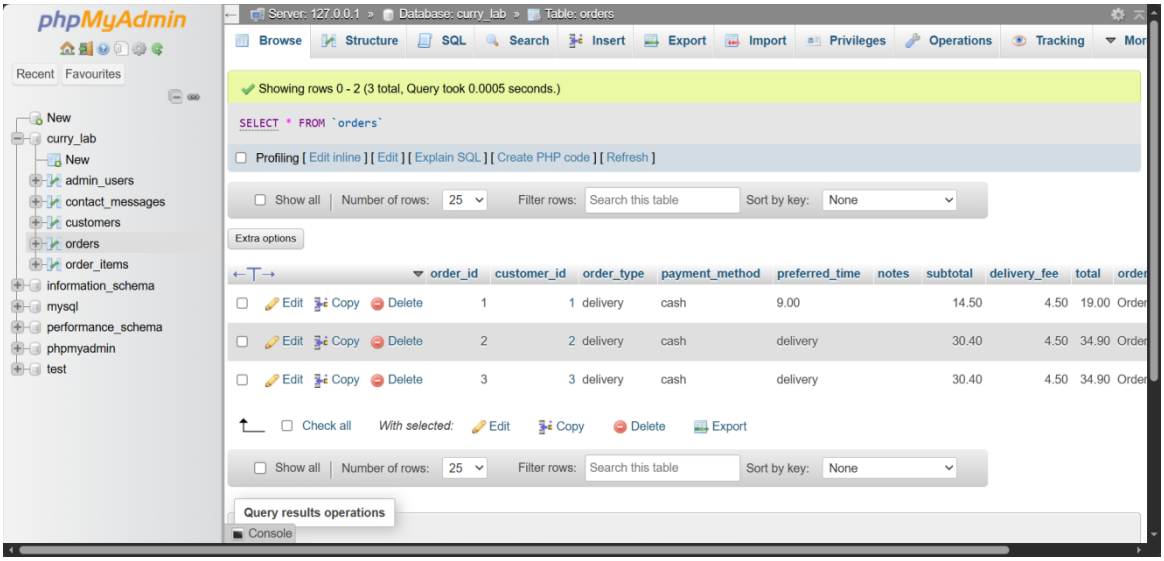
Figure 26. Order Submission message



The test also determines whether the delivery option behaves properly. With the pickup option selected, the address field is optional. When the delivery option is selected, the address fields become mandatory. When the card option is selected, the fields for credit card information become visible. These are the results of the tests indicating proper behaviour.

Figure 27 shows that the test for the backend was performed by placing an order and determining whether this order was available in the database and admin page. This indicates that the frontend and backend have been successfully combined.

Figure 27. Submitted order visible in phpMyAdmin



To ensure the reliability and correctness of the developed system, functional testing was conducted for all major user interactions. The entire ordering workflow, including menu selection, cart operations, delivery form behaviour, payment selection, order submission, and administrative order viewing, was validated during the testing process. Each test case was created to confirm whether the system operated as intended under typical user circumstances. The comprehensive functional test cases used to assess the system are shown in Table 3

Table 3. Test case table

Test Case ID	**TC 01**
Test Scenario	Add item to cart
Test steps	Navigate to menu → Click “Add to Cart”
Test data	Food item : Butter Chicken
Expected result	Item is added to cart and stored in local storage
Actual result	Item successfully added
Status	Pass
Evidence	Figure 12, Figure 13 refers to, where customers can see all the selected foods before moving to the checkout process. This page accesses the data of the cart stored in the local storage of the browser and shows all the selected foods together with their prices and quantities along with totals per each selected food. Figure 13. Cart page

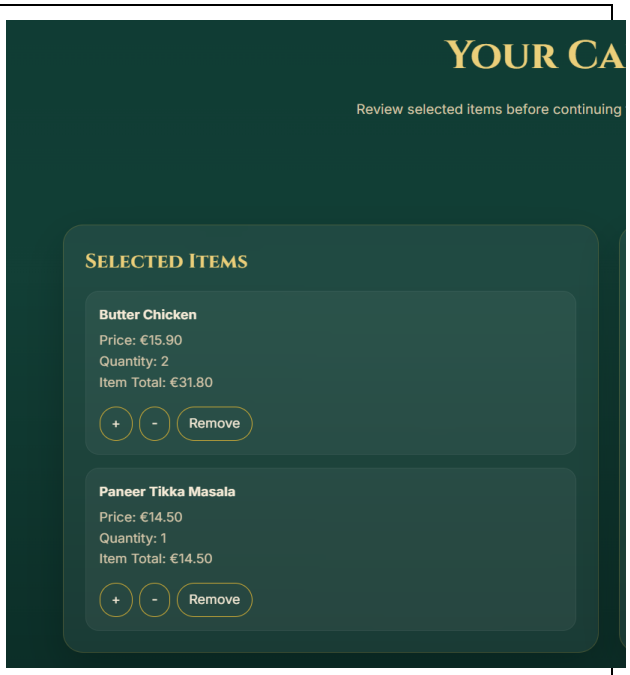
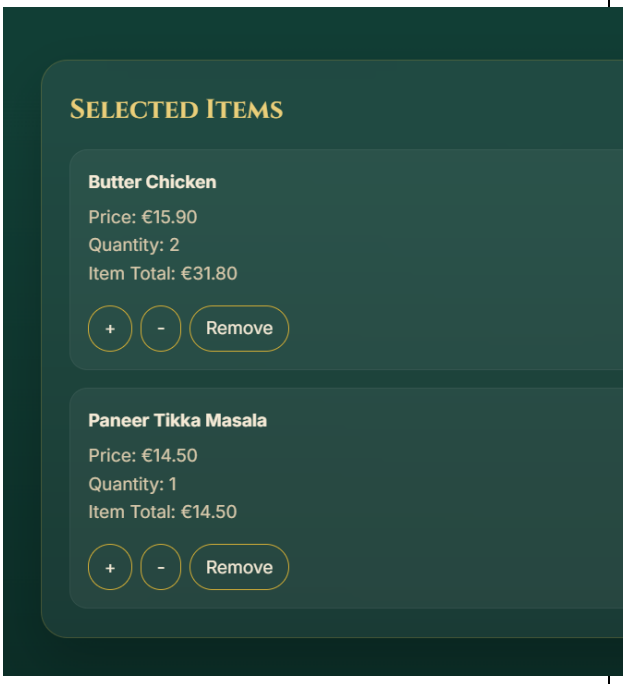


Figure 14 illustrates that add, reduce, and delete functions ensure that the shopping cart reacts dynamically. In the event of any change in quantities, the cart is stored locally and reloads itself. This confirms real-time updating of the cart without reloading the page.

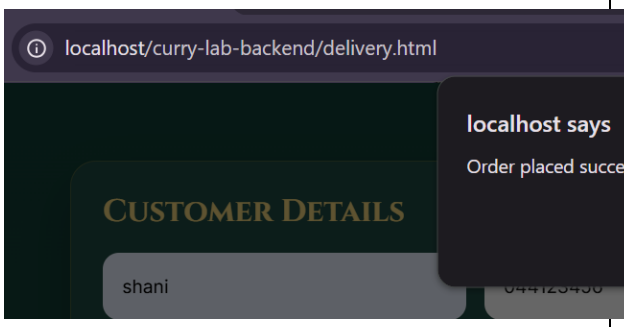
Figure 14. Cart - selected Items



The cart page consists of two parts: selected

	items and an order summary. Selected items refer to the items in the cart, whereas the order summary in Figure 15 refers to the subtotal and total amount of the purchase. Navigation buttons are present on the page to direct users to either the menu or the delivery page. The functionality of the cart is coded in the cart.js file. The loadCart() method is used to obtain cart data, remove the current data from the page, calculate the subtotal value, and create cart items dynamically. If the cart contains no products, a message is shows to inform the users.
Test Case ID	**TC 02**
Test Scenario	Update item quantity
Test steps	Increase/decrease quantity in cart
Test data	Quantity +1/ -1
Expected result	Cart updates and total recalculates correctly
Actual result	Total updates correctly
Status	Pass
Evidence	Figure 14
Test Case ID	**TC 03**
Test Scenario	Delivery selection validation
Test steps	Select “Delivery option”
Test data	Address fields empty
Expected result	Address fields become required
Actual result	Fields required correctly
Status	Pass
Evidence	Figure 16
Test Case ID	**TC 04**
Test Scenario	Pickup selection validation

Test steps	Select “Pickup” option
Test data	Address fields filled
Expected result	Address fields cleared and not required
Actual result	Fields disabled correctly
Status	Pass
Evidence	**Error! Reference source not found.**
Test Case ID	**TC 05**
Test Scenario	Card payment visibility
Test steps	Select “ Card payment”
Test data	Payment method = card
Expected result	Card fields appear dynamically
Actual result	Fields displayed correctly
Status	Pass
Evidence	Figure 18
Test Case ID	**TC 06**
Test Scenario	Order submission
Test steps	Fill form → click “place order”
Test data	Valid form data
Expected result	Order saved in database
Actual result	Order stored successfully
Status	Pass
Evidence	To test the system, a procedure for the complete ordering process was performed starting from the menu page to the admin order viewing page. First, the user accesses the menu page, where the customer selects food items and ensures that these food items have been placed inside the cart successfully as shown in Figure 26. The cart page is then accessed again to check whether the selected items, quantity, subtotal, and total

	have been displayed accurately. Figure 26. Order Submission message  The test also determines whether the delivery option behaves properly. With the pickup option selected, the address field is optional. When the delivery option is selected, the address fields become mandatory. When the card option is selected, the fields for credit card information become visible. These are the results of the tests indicating proper behaviour. Figure 27 shows that the test for the backend was performed by placing an order and determining whether this order was available in the database and admin page. This indicates that the frontend and backend have been successfully combined. , Figure 27
Test Case ID	**TC 07**
Test Scenario	Admin order view
Test steps	Login → open orders page
Test data	Existing orders
Expected result	Orders displayed in table
Actual result	Order visible
Status	Pass

Evidence	Figure 23

The test case outcomes show that all essential functions of the system operated as intended. The payment area reacts appropriately to the chosen options, the delivery form is dynamically modified based on the user input, and the cart actions reliably update the numbers and totals. Additionally, the successful database storage of orders and their visibility in the admin panel attest to the correct frontend and backend component integration. These outcomes show that the system can handle the planned ordering procedure and is functionally reliable.

Although the system passed all functional test cases, the testing was limited to controlled conditions and did not include stress testing or security testing. Future improvements should include validation against invalid inputs, protection against malicious data, and performance testing under high loads.

The implementation and assessment of the curry Lab web-based meal ordering system are covered in this chapter. The system offers a comprehensive ordering workflow by integrating frontend and backend technologies. The practical application of web development concepts is demonstrated using database integration, dynamic interaction, and modular architecture.

Although not entirely production-ready the system offers a solid basis for future development. The outcomes that the system achieves its goals and show that an independent online ordering platform is feasible.

6 Results and conclusion

The following are the results achieved after developing and implementing the Curry Lab online order processing system. This chapter will analyse the effectiveness of the system with respect to achieving the research objectives and its performance in terms of functionality, usability, and design. In addition, some of the limitations experienced in the process will be outlined, and suggestions for improvement will be made.

6.1 Summary of results

The developed system provides a full web-based ordering process that starts from browsing through the menu until the management of the orders via the administration panel. The results show that all functionalities were successfully implemented and tested.

Through a structured menu interface, the system enables users to browse food products, choose items, and manage them in a dynamic cart. Real-time updates, such as quantity modifications and total price computations, are supported by the cart capability. The checkout procedure gathers user information, dynamically adjusts to delivery or pickup choices, and controls delivery costs. Conditional input fields improve usability by ensuring that users only interact with pertinent form elements.

Additionally, order data can be stored in a relational database owing to backend integration. The ability of the system to manage data durability and retrieval was confirmed by the successful appearance of submitted orders in both the database and the admin interface. A minimum level of security is added to the system by implementing an admin login system, which guarantees that only authorised users may access sensitive order data. Overall, the system satisfies its functional criteria and shows that an independent food ordering platform for a small restaurant is feasible.

6.2 Evaluation of system objectives

The aim of this project was to develop an online food ordering system, that minimises dependency on external third-party systems. The findings of the implementation and tests conducted indicate that this aim has been achieved.

This system offers a direct order placement facility wherein orders are placed without using any other third-party platform. This supports the objective of avoiding dependence on commission-based systems. It also provides companies with more freedom to manage their business activities. Moreover, the use of a backend database system helps to store and access orders.

The system performed well in terms of usability. The methodical sorting procedure minimises user confusion, and the navigation structure is unambiguous. A seamless user experience is enhanced by dynamic elements including payment method selection, delivery form modifications, and automatic cart updates.

However even as the system satisfies its main objectives, there is much about it that does not match the features of commercial software. The ability to track orders in real time, online payment processing, and user account systems are some of the missing elements. Clearly, the system can only serve as a prototype or pilot project.

6.3 System performance and usability evaluation

The application shows impressive capabilities in terms of speed and interactivity. All front-end processes such as menu navigation, shopping cart additions, and form processing are performed effectively using client-side JavaScript. The employment of local storage ensures that shopping cart information remains intact even when changing web pages without interacting with the server.

Backend processing is effective within the scope of the project. Requests are transmitted from the front-end to the back-end through asynchronous requests without problems. The information was saved in the database without errors. Organised queries and database relations help organise information and make it easily accessible.

The system's responsiveness, ease of use, and clarity of the interface design were considered and evaluated for usability. The user experience is enhanced by using a common visual style button labels and organised layouts. By removing unnecessary input and directing user engagement, conditional form fields further enhance usability.

Although there are several merits, there is room for improvement in terms of its usability. For instance, there is no feedback from the users, aside from the general notifications, and there is no confirmation page for the order placed. placed.

6.4 Limitations of the system

Although the system creates an operational ordering system, some constraints arise. One such constraint is the lack of payment processing capability. The system merely shows a selection of payments but does not carry out the actual payments.

Another limitation is the fundamental security feature incorporated into the system. Even though the admin login feature limits access to order information, it does not have any password encryption process or enhanced security features. Therefore, there may be security threats when implementing the system in practice.

User account capabilities are another feature that is missing in the current system design. Users cannot establish user accounts review their previous orders or save preferences. This will have a negative impact on the system because it makes it less personalised than modern customers would expect in an ordering platform.

In addition, performance testing under high user loads has not been performed. The system has not undergone performance testing to evaluate its scalability and concurrent users.

6.5 Recommendation for future improvements

Several enhancements can be suggested for further development based on the restrictions that have been found. The incorporation of a safe online payment method is one of the most significant improvements. This would enable the system to handle actual transactions and qualify it for commercial use.

The use of sophisticated security features, such as safe authentication procedures, password hashing, and defence against prevalent online vulnerabilities, is another enhancement. These steps would greatly improve the dependability and credibility of the system.

The inclusion of features such as customer registration, login, and order management would make the system more efficient. Such functionalities would include order history; stored addresses, and even personalised recommendations for users. Another useful

feature is the real-time tracking of orders. This would help customers know the development made in the order process.

Finally, performance optimisation and scalability tests must be performed to determine whether the system is capable of handling more users. This is important when implementing the system.

6.6 Conclusion

This thesis focuses on an affordable web-based ordering and delivery management system for small Finnish food companies. It should be mentioned that the research is focused on the practical problems of small food companies, their dependency on third-party ordering services characterised by high commissions, inflexibility, and lack of data ownership. To resolve these issues, this thesis provides answers to three research questions concerning system requirements, development methods, and autonomy.

The first research question explored how the technical and functional specifications of a web-based ordering and delivery management system for small food delivery companies can be determined. The findings from the literature review and system analysis revealed that an effective system for small businesses should focus on core functionalities rather than unnecessary complexity. Key functional requirements were established, including browsing the menu, cart management, customer details management, placing an order, managing deliveries and pickups, and monitoring orders by administrators. In terms of technical aspects, a three-tier architecture incorporating frontend, backend, and database tiers proved to be the right choice owing to its simplicity, ease of maintenance, and scalability for small business purposes. The selected technologies including HTML, CSS, JavaScript, PHP, and MySQL were deemed appropriate, because of their affordability and effectiveness in executing all the necessary system operations.

The second research question investigated how to create an efficient and effective online ordering system using the Rapid Application Development methodology and adequate system architecture. The findings reveal that the RAD methodology was extremely successful in terms of development. The system continued to improve through iterative prototyping and continuous enhancement, resulting in higher usability and efficiency. As a result of the application of the concept of usability, the system became extremely user-friendly. Features such as the dynamic updating of the shopping cart, conditional delivery

and pick-up forms, good navigation, reusable code and elements, and responsiveness made the web site user-friendly. In addition, the three-tier architecture of the system helped distinguish between the user interface, application logic, and database management, which made the system efficient and expandable.

The third research question addressed the issue of improving the ordering process and reducing dependency on outside logistics services for smaller businesses. The developed system has successfully proven that the ordering process can be managed by an independent platform, allowing businesses to gain more control over their operations and minimising the use of third-party services. While working differently from third-party ordering sites, the created system allows businesses to control their orders, maintain full ownership of their client data, avoid commission prices and customise the functionality of the site according to their needs. Integration with the relational database allowed the storage and retrieval of all orders, and the admin panel allowed effective order management. Despite the inability to substitute the logistics capabilities of commercial websites, the system provides a convenient solution for smaller businesses.

The implementation and testing results confirmed that the developed prototype successfully achieved its intended objectives. Functional testing validated that major features, including cart operations, delivery validation, order submission, backend storage, and administrative order viewing, operated correctly. These findings show that a reliable and functional ordering platform can be developed using low-cost technologies without the need for highly complex infrastructure.

Despite these successful outcomes, several limitations remain. Advanced functionalities such as secure payment gateway integration, real-time GPS tracking, customer accounts, enhanced cyber security mechanisms, and large-scale performance testing were not included because of the scope limitations of a bachelor-level thesis. These limitations suggest clear directions for future research and development.

Finally, it is possible to note that this thesis answered all the three research questions proposed, proving that it is possible to develop a cost-effective web-based order management platform that will be able addresses the operational requirements of small food enterprises. This research makes an important contribution not only on a theoretical level but also on a practical one because it provides an example of how affordable IT technologies and usability-oriented approaches can be used to create an independent platform.

7 Summary

This thesis aimed to create a web-based ordering and delivery management system specifically designed for small food delivery enterprises. It examined the problems that small restaurants face when using third-party food delivery websites such as high fees, lack of control, and inability to access information about customers.

The theoretical part of the research covered the literature on online food order systems, RAD methodology, usability considerations, system design, and digital transformation in small firms. A knowledge gap was identified in the field of simple and inexpensive digital products for small-scale food delivery services.

The practical section included designing and developing a sample ordering system using HTML, CSS, JavaScript, PHP, MySQL, and XAMPP. The system architecture contains a three-tiered model incorporating front-end, back-end, and database layers. Some of the system functionalities were menu browsing, managing the shopping cart, checkout process, order placement, database connectivity, and order management by the administrator.

Tests were performed to establish the usability and reliability of this system. The functionalities worked well, such as the selection of items, modification of the shopping cart, validating deliveries, filling out the payment form, and storage of orders on the back-end side of the system. This project clearly highlighted the capability of a stand-alone ordering system to boost efficiency and decrease reliance on outside delivery services among small enterprises.

Despite the lack of more sophisticated commercial functions such as payment portals and real-time tracking, the main goal was accomplished, namely designing an affordable and effective ordering system. It can be concluded that independent ordering systems can bring many advantages to small catering establishments and have great potential for improvement.

References

Chavan, V., Jadhav, P., Korade, S., & Teli, P. (2015). *Implementing Customizable Online Food Ordering System Using Web Based Application*. *2*(4).

Egereonu, S. K. (2024). *Optimized Web-based Online Food Ordering System: Design and Implementation*. *2*(2).

Fred D., D. (n.d.). *Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology*. Retrieved https://www.researchgate.net/publication/200085965_Perceived_Usefulness_Perceived_Ease_of_Use_and_User_Acceptance_of_Information_Technology

Hossain, Md Fahad. (n.d.). Retrieved 17 April 2026, from https://ar.iub.edu.bd/bitstream/handle/11348/776/2022%20Summer%2c%201730416%2c%20Online%20Food%20Ordering%20and%20Restaurant%20Management%20System.pdf?sequence=1&isAllowed=y

Huang, Y., & Khantong, P. (2024). INFLUENCE OF PRICE STRATEGY ON CUSTOMER SATISFACTION IN RESTAURANT BUSINESS. *Procedia of Multidisciplinary Research*, *2*(10), 59–59. https://so09.tci-thaijo.org/index.php/PMR/article/view/5356

IBM. RAD. (2024, November 25). https://www.ibm.com/think/topics/software-development

IBM. Three-tier architecture. (2021, October 18). https://www.ibm.com/think/topics/three-tier-architecture

Jesse James, G. (2011). *The Elements of User Experience: User-Centered Design for the Web and Beyond, Second Edition*. https://ptgmedia.pearsoncmg.com/images/9780321683687/samplepages/0321683684.pdf

Mozilla Developer Network. (2025, April 11). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/First_steps

NebiOğlu, O. (2020). Review of Menu Management Process Model with A Case Study. *Advances in Hospitality and Tourism Research (AHTR)*, *8*(2), 203–234. https://doi.org/10.30519/ahtr.668220

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann.

OECD. (2023). OECD. https://www.oecd.org/en/topics/digital.html

Oghenekaro, L. U., & Okafor, J. C. (2023). Web-Based Integrated Restaurant Management System. *International Journal of Applied Information Systems*, *12*(40), 48–53. https://doi.org/10.5120/ijais2023451945

Piyatissa, W. B. A. C. (2020). *Web Based Restaurant Management System* [A dissertation submitted for the Degree of Master of Information Technology]. University of Colombo School of Computing. https://dl.ucsc.cmb.ac.lk/jspui/bitstream/123456789/4501/1/2017%20MIT%20055.pdf

Senarath, U. S. (2021). *Waterfall Methodology, Prototyping and Agile Development*.

Wang, X. (n.d.). *Bachelor's Programme in Software Engineering, Bachelor's thesis*.

World Bank. (2022). https://www.worldbank.org/ext/en/topic/competitiveness/small-and-medium-enterprises-smes-finance

Appendix 1: Data management plan (this is must for all)

Description of thesis research data

This thesis focuses on the design and development of a cost-effective web-based ordering and delivery management system for small food businesses. The research data used in this thesis mainly consisted of academic literature, system development files, database structures, screenshots, and functional testing records. The literature review data were collected from journal articles, books, conference papers, and official documentation websites related to web development, online food ordering systems, usability, and digital transformation.

The practical development data include HTML, CSS, JavaScript, PHP, SQL files, Entity Relationship diagrams, system architecture diagrams, and testing evidence generated during the implementation of the system. Functional testing was conducted by simulating customer orders, cart operations, delivery selection, and administrative order management. The test data were created only for demonstration and educational purposes. No real customer or commercial business data were used in this study. The research data formats included text documents, source code files, screenshots, SQL database files, and image files.

Management and storage of the research data

Thesis data and files used to develop the project were stored in the author's password-protected computer for safety and confidentiality purposes. The project was developed using the XAMPP local server environment, meaning that the project and its database were run only locally on the author's computer. Copies of the key project files were backed up in cloud storage and on external storage media for data preservation.

Access to the project material was limited to the thesis author and thesis supervisor in situations where the supervisor needed to evaluate or assess the project. The source code, screenshots, database files, and documentation files were arranged in different folders for ease of access. All data handling involving the research was performed based on the GDPR guidelines and best practices in research.

Processing of personal data and sensitive data

No actual personal or sensitive data belonging to customers were collected or processed during the development of this thesis. All order data, names, addresses, phone numbers, and other customer data

that were utilised in the tests were purely fictional sample data and were generated only for educational and demonstration purposes. Thus, the risk of data leakage was low.

Nevertheless, GDPR standards were also considered while working on the prototype of the online shop. In particular, principles such as data minimisation, confidentiality, security of stored data, and restriction on access, were taken considered. However, if this system is implemented commercially in the future, more actions should be taken according to GDPR standards.

Ownership of research data

The research data, source code, system documentation, screenshots, and implementation materials developed during this thesis are owned by the author. Academic references and external literature sources remain the intellectual property of their original authors and publishers and were used only for educational and citation purposes in this thesis.

The developed prototype system and project materials may be used by the author for educational, portfolio, and future development. Project data will not be commercially distributed without proper authorisation.

Further use of research data after the completion of the thesis

After the completion of the thesis, the final thesis report, source code files, database structures, screenshots, and related project materials will be securely stored by the author for future educational and professional portfolio use. These materials may also support future improvements and the expansion of the developed system.

Temporary testing files, duplicate backups, and unnecessary demonstration data may be securely deleted after the thesis evaluation process is complete. Since no real personal or sensitive data were collected during the research, no additional data anonymous procedures were required after the completion of the thesis.