Restaurant Management System

By
Asif Karim
193-15-13521

FINAL YEAR DESIGN PROJECT REPORT

This Report Presented in Partial Fulfillment of the Requirements for the **Degree of Bachelor of Science in Computer Science and Engineering**

Supervised by

Mr. Md. Abbas Ali Khan
Assistant Professor
Department of Computer Science and Engineering Daffodil International University

Co-Supervised by

Md. Abdullah Al Kafi
Lecturer
Department of Computer Science and Engineering Daffodil International University

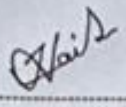
DAFFODIL INTERNATIONAL UNIVERSITY
Dhaka, Bangladesh

May 14, 2025

APPROVAL

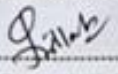
This Project titled "**Restaurant Management System**," submitted by **Asif Karim**, ID No: **193-15-13521** to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on **14-05-2025**.

BOARD OF EXAMINERS



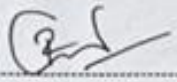
.....

Dr. Sheak Rashed Haider Noori — **Chairman**
Professor and Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University



.....

Md. Masum Billah — **Internal Examiner**
Assistant Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University



.....

Mr. Partha Dip Sarkar — **Internal Examiner**
Lecturer
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

.....

Dr. Md. Zulfiker Mahmud — **External Examiner**
Professor
Department of Computer Science and Engineering
Jagannath University

©Daffodil International University

DECLARATION

We hereby declare that this project has been done by us under the supervision of **Mr. Md Abbas Ali Khan, Assistant Professor**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

Mr. Md. Abbas Ali Khan
Assistant Professor
Department of Computer Science and Engineering Daffodil International University

Co-Supervised by:

Md. Abdullah Al Kafi
Lecturer
Department of Computer Science and Engineering Daffodil International University

Submitted by:

Asif Karim
Student ID: 193-15-13521
Department of Computer Science and Engineering Daffodil International University

©Daffodil International University

ACKNOWLEDGEMENTS

This work would not have been possible without the support and contributions of many individuals over the past two semesters. We are deeply grateful to everyone who has assisted us in one way or another.

First, we express our heartfelt thanks and gratefulness to the almighty for His divine blessing making it possible for us to complete the **Final Year Design Project(FYDP)** successfully.

We are grateful and wish our profound indebtedness to **Mr. Md. Abbas Ali Khan**, **Assistant Professor**, Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh. Deep knowledge and keen interest of our supervisor in the field of **Web Development** to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting themat all stages have made it possible to complete this project.

We would like to express our heartfelt gratitude to the Head of the Department of Computer Science and Engineering, for his kind help in finishing our project and also to other faculty members and the staff of the Department of Computer Science and Engineering, Daffodil International University.

We would like to thank our entire course-mates at Daffodil International University, who took part in this discussion while completing the coursework.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

ABSTRACT

The project involves developing a restaurant management system using Laravel, designed to help with the everyday tasks at small to mid-sized restaurants. The system was designed to allow customers to view the menu, book places at tables and talk to the restaurant. Behind the scenes, the restaurant admin can use the secure dashboard to manage all aspects of the menu, group of chefs and reservation system. Because local restaurant operators still depend largely on manual tasks that are usually time-consuming and confusing, this system was designed. The project used the software development life cycle (SDLC) and was planned, analyzed, designed, developed and tested one step at a time. Thanks to its eloquent ORM and in-built authentication, the system could be built with ease and is reliable. Trough which is being tested locally on a server, is working fine and it will help build a foundation for future developments like notifications or live deployments. Throughout this project, we explored ways to use today's tools from web development to tackle real-world challenges in an efficient manner.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Introduction

In today's fast-paced digital environment, restaurants that depend entirely on traditional methods often struggle with delays, miscommunication and missed business opportunities. This project introduces a web-based restaurant management system aimed at helping restaurants manage daily tasks more effectively and professionally. Customers can easily browse the menu, reserve a table, or send a message, while restaurant owner can handle reservations, update content, and stay organized using a simple admin dashboard. The system is especially helpful for small and medium-sized restaurants that want to modernize their services without spending heavily on expensive commercial software. By focusing on practical features and user-friendly design, this project aims to make restaurant operations smoother while meeting the rising expectations of today's digital-savvy customers.

.

1.2 Motivation

It was obvious, whether I visited restaurants or saw them on the internet, that a lot of small businesses still use outdated ways to handle booking, menus and contacts with customers. Because of this, customers may be confused, patiently wait for answers or switch to another company. In addition, most people now expect that using digital tools for ordering or making a reservation is an effortless process. Having expensive and difficult-to-use solutions is the main issue for small restaurants today. Because customers were expecting more than the systems we had, I decided to make a system that is useful and still economical. I wanted to offer an easy way for both customers and restaurant managers, so I used tools like Laravel and MySQL for developing the solution. Since I used my technical ability to help solve a practical business issue, the project made a difference in daily business tasks.

1.3 Objectives

The main aim of this project is to produce a system that can meet the needs of customers and restaurants. The targets set out are:

1. To make a user-friendly and convenient web application for menus, making reservations and communicating with the restaurant.
2. Deliver an admin area that is easy for restaurant owners to use when adding, editing and changing menu, chefs and reservations information.
3. Design the backend of an application using the Laravel MVC structure to ensure that it is secure and able to handle a growing number of requests.
4. To use MySQL in a way that makes organizing and storing data convenient.
5. To establish safe authentication for securing admins and their associated data.
6. To make record-keeping easier and smoother with the help of computers and improve interactions with customers.
7. To design a computer system that is straightforward to update, extend and use in a real restaurant.

1.4 Methodology

To develop the Restaurant Management System, I followed a structured and hands-on approach based on the Software Development Life Cycle (SDLC). Each stage was carefully planned and implemented to ensure the system would be both functional and user-friendly. Here's how the development process unfolded:

1. Requirement Analysis: I began by figuring out the requirements from the customers and restaurant admins. These features included making reservations, looking at the restaurant menu, introducing the chef and content control for the app. I also researched related systems which allowed me to simplify the list of main features for applications in small and medium restaurants.
2. After the requirements were understood, I began designing the structure and organization of the system. I used wireframes to design a layout for the part of the app shown to customers and for the part used by admins. I also made sure that the database was organized so information could be handled smoothly in every section of the system.
3. The development was divided into two parts: developing the frontend and the backend. Frontend: A clean, responsive and user-friendly interface was created using HTML, Tailwind CSS and Laravel's Blade technology. • I was able to organize everything in Laravel thanks to its MVC framework. Controllers

controlled the program, models linked to the database and views showed the results. I set up secure page access with middleware and organized page routes using Laravel. • All the data such as menu, chefs, reservations, categories and messages from customers, were managed with MySQL.

4. Manually testing: I went over all the parts of the system during development to see if they functioned properly. I checked different routes, worked on form validation and operated on the database. I could quickly address and solve small problems with the help of Laravel.
5. I tested the system on my computer using XAMPP, but I also prepared it for later use in the cloud. The project was set up using environment files, organizing database migration and folders so deploying it online will be straightforward later.

1.5 Project Outcome

The platform created with restaurant management software is now reliable, flexible and helps streamline restaurant tasks. The project fulfilled each of its important targets, letting users enjoy a simple frontend and boasting an administrator dashboard with advanced features. Here is a short summary of what was decided.

1. An improved way for customers: Anyone checking the website can quickly view the menu, discover the chefs and fill out a reservation online.
2. Secure Panel for Restaurant Staff: With a clean and secure back panel, staff can easily handle menu items, chefs, categories and the homepage sliders.
3. When data is stored in a well-organized database, content changes are applied immediately and you do not need to modify the coding.
4. Laravel's authentication ensures that only relevant administrators have access to the login system.
5. The design of the system is flexible, making it easy to upgrade later on by adding online ordering, allowing customers to log in or creating reports.
6. Deployment: All configurations for moving the application to a live environment are already complete and ready to go.

1.6 Organization of the Report

Chapter 1: Introduction

The purpose of this chapter is to explain the reason we need the Restaurant Management System. It outlines the issues that many small and mid-sized restaurants encounter while operating on a daily basis and shares how the project aims to supply a digital answer. The objectives are outlined, the main methodology is discussed and the remainder of the report is described.

Chapter 2: Background and Literature Review

In this chapter, I provide a background and review the relevant literature. This section of the report observes the current solutions in restaurant management and outlines the limitations linked to their costs and complexity. In addition, it highlights why Laravel was chosen, explaining its ability to be flexible, safe and user-friendly. The chapter utilizes information from both research and business practice to emphasize that small businesses require a more accessible system.

Chapter 3: System Design and Specification

Looks at the process of designing and defining a system. This chapter discusses in detail how the project is built. Describes how the program is built using the MVC architecture and details the structure of the database system. It creates modules for various parts, both at the frontend and backend. The list contains what the system should be able to do and how the system should carry out those functions.

Chapter 4: Implementation and Results

The findings are demonstrated and discussed. Here, the report focuses on carrying out the plans put in place earlier. It includes information on how the system was developed, what tools were important and how testing was carried out to ensure nothing was broken. It also provides essential features such as handling reservations, handling content and displaying profiles. I have also included several screenshots that demonstrate how the application works. Creating something that is user-friendly and will respond well on any device is also discussed in development challenges.

Chapter 5: Engineering Standards and Design Challenges

The section discusses the software development guidelines, covering code quality, handling different versions of the code and security of data. It also tells us about how designers strike a balance between usefulness and simplicity. It is explored how a digital menu system can benefit small dining businesses without requiring them to spend a lot of money.

Chapter 6: Conclusion

The final part of the report assesses how the objectives were achieved and explains what the system provides. Still, it points out advantages and makes suggestions. On the whole, the chapter points out the usefulness of the system for our everyday experiences.

Chapter 2
Background

2.1 Introduction

To appreciate the purpose of the project, you should learn about the current methods used by restaurants. Even today, a lot of small eateries continue to do tasks such as reservations and updating menus using old-fashioned methods. Both large and small restaurants can use restaurant software, but the costly features offered to big chains are too much for some small businesses. As technology advances in hotels, even small and medium-sized dining establishments are searching for new ways to handle their daily work. Although tools are popular in this industry, most independent restaurants find they are too costly and complicated to use. Our goal is to create an easy-to-use and affordable Restaurant Management System (RMS) that is built with Laravel. Restaurant owners use the system to control their list of dishes, accept reservations and communicate with customers on one platform that is simple and suitable for both small and large businesses. In this chapter, we will explore what tools are currently available for restaurants, review the technology behind this project and mention the problems this project hopes to solve. Gaining a better understanding of this background will explain why small restaurants need this project and how it hopes to help them.

2.2 Literature Review

Table 2.1: Summary of Literature Reviewed.

Author (s)	Year	Title	Methodology	Key Findings
Raj, M., Sundararajan, A., & You, C.	2020	COVID-19 and Digital Resilience: Evidence from Uber Eats	Qualitative Analysis	Small restaurants experienced significant increases in activity on the Uber Eats platform during the pandemic, highlighting the importance of digital channels for business resilience
Trikoyat, Y., Yulianto, & Maria, E.	2022	Information System of Muara Badak Village Culinary Sales Using Laravel Web-Based	Case Study	Developed a web-based culinary sales system using Laravel, demonstrating the framework's suitability for small-scale restaurant applications.
S. N. Cheong, W. W. Chiew, W. J. Yap	2010	Design and Development of Multi-touchable E-restaurant Management System	Quantitative Analysis	Presents the design and development of a multi-touchable e-restaurant management system, aiming to enhance the dining experience through technology.
Patel, K., & SevenRooms	2023	Data Rich, Data Dumb': Can AI Save Struggling Restaurants?	Industry Report	Discussed how AI can enhance efficiency, reduce costs, and personalize customer experiences in the restaurant industry.

2.2.1 Similar Applications

1. **Engaze**: Engaze provides cloud-based services to over 1,000 restaurants in Bangladesh. Restaurants can use it for online orders, updating their menus and monitoring performance which optimizes their business. It allows workflows at a restaurant to be more efficient and also helps promote their business using social media.

   Key Features:

   1. Cloud-based POS stands for Point of Sale system.
   2. Organizing and analyzing orders
   3. Use social media sites for promoting your business.
   4. Managing and following products used

2. It is widely used for managing restaurants in Bangladesh. Users can take care of billing, menu and inventory management in real time using the app. Restaurants that need a straightforward tool with many features can rely on Posist. •

   Key Features:

   1. The order and billing management system manages everything.
   2. The software provides instant updates on your stock and alerts when needed.
   3. Incorporation of platforms that allow customers to order online
   4. Multi-location support is available through the cloud.
   5. All items on the menu and their pricing managed from one place.

   Just like the proposed Restaurant Management System, both of these applications allow users to handle orders and change the menu. While this design might seem similar, it sets itself apart with secure signing in, a reliable architecture and a dedicated area for management which makes upgrading and adding functions easier in future.

2.2.2 Related Research

Researchers have studied a variety of aspects of restaurant management systems. A particular study mentions that by using IoT and cloud computing, it is now possible to keep track of stocks and sort orders instantaneously. Another study emphasizes reserving, ordering and communicating using mobile applications to enhance both how quickly the service is provided and how guests feel about it. There is also research that studies the ways cloud-based offerings help restaurant businesses cut their expenses, keep their systems clean and simple and secure their customers' information through encryption. A review of restaurant management systems analyzes the progress made in managing inventory, making reservations and using AI for making predictions. All in all, these outcomes drive the shift toward improved security and better customer service for restaurants.

2.3 Gap Analysis

Features	Engaze	Posist	Proposed system
Online table reservation	No	Yes	Yes
Chef Profile Management	No	Yes	Yes
Real-time Admin Dashboard	Yes	Yes	Yes
Secure Customer Authentication	Yes	Yes	Yes
Reservation System	Yes	Yes	Yes
Contact Form	Yes	Yes	Yes
Responsive Design	Yes	Yes	Yes
Multi-language Support	No	No	Yes
Recommendations or filtering latest products	Yes	Yes	Yes
Product add to cart	Yes	Yes	Yes
Admin Notifications	Yes	Yes	Yes

Table 2.3: Gap Analysis

Most of the restaurant systems available cater to companies operating at large scales. Since most of these platforms come with sophisticated features, they are rarely used by small or mid-sized restaurants. It can be challenging for small businesses to pay for monthly fees, learn the software and purchase the necessary hardware. There are many options out there, but most don't fit local restaurants since they are either overly simplistic or have many unnecessary features. Certain platforms are not flexible enough for business owners to modify the system according to their real business process. It aims to close that

gap by introducing a simple and low-cost answer created for small restaurants' main requirements. Thanks to Laravel and new web technologies, the system offers both the admin dashboard and customer interface which are practical and enjoyable, without any features you don't need. The desired outcome is to give restaurant owners more power and a simple solution, as such tools are rarely available today.

2.4 Summary

This chapter discussed the history of restaurant management systems and the move from using paper systems to digital options. It looked into platforms like Engaze and Posist, looked into newer technologies like Laravel and presented information from studies involving similar platforms. During our analysis, we found that most of the current systems either price small to mid-sized restaurants out of their budget or are too complicated. For that reason, our proposal creates a system that helps smaller companies with an easy-to-use, inexpensive and customizable set of tools. The information we covered shows the relevance and necessity of the project to modern restaurants.

Chapter 3

Requirement Analysis

3.1 Requirement Analysis & Design Specification

3.1.1 Overview

Prior to beginning the development process, both the customers' and the restaurant workers' requirements were analyzed thoroughly. The purpose was to understand the features and functionality the system should have when used in real situations. To help with the design, I looked into other similar platforms and asked for input on table reservations, handling chefs' profiles, updating the menu and the safe admin dashboard. Thanks to this analysis, the main features of the system could be outlined and the final product would suit the needs of the users.

.

3.1.2 System Design

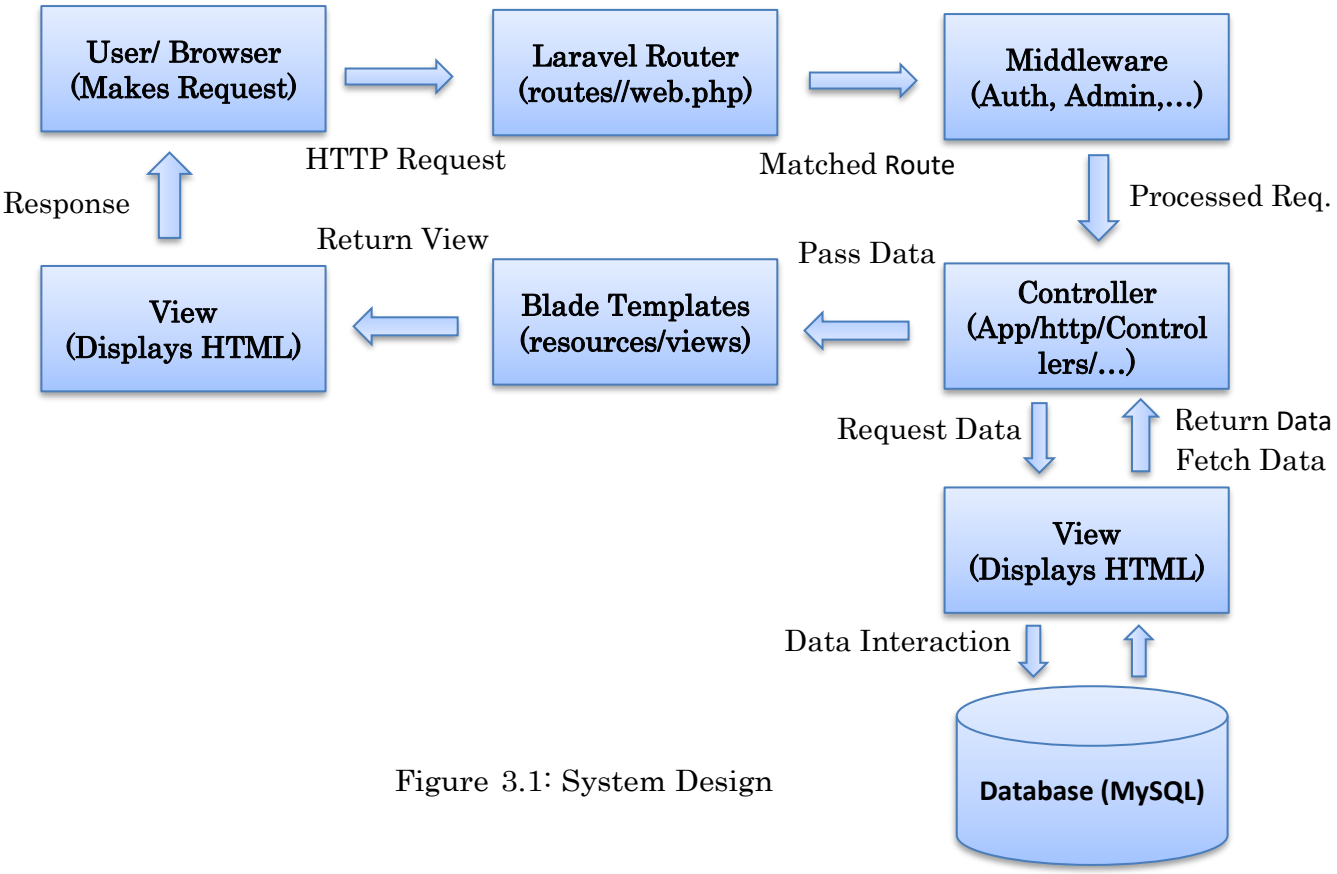


Figure 3.1: System Design

3.1.3 Functional and Nonfunctional Requirements

This In this part, we explain what must be included in the Restaurant Management System by function and nonfunction. A system's functional requirements tell it what to do, while the nonfunctional ones outline how to behave so that everything runs smoothly for both users and administrators.

Functional Requirements To fulfill its main aims, the system should include the functional requirements that describe important features and actions. This list of requirements allows the system to perform as required and gives users the features to supervise their restaurant activities.

1. Registration and Login: All users (customers and admins) should be able to register securely and logged into their accounts. Administrators will have special access to manage the data on the platform.
2. Admins should have access to management of the restaurant's menu and use it to add or remove items, assign them categories, insert price tags, provide descriptions and add images.
3. Online Reservations: Individuals should have the option to schedule a time and date online and this information is confirmed for both them and the restaurant.
4. A feature should be provided so that customers can send inquiries to the support team. Employees should be able to check and respond to messages from the admin dashboard.
5. Admins should be allowed to build chef profiles with photos, names and their various specialties.
6. Administrators should be able to update the slideshow on the homepage to promote happenings, special offers or important items.
7. Reservations Management: Admins need to be able to view all reservations and should confirm or make changes to them as required.
8. Whenever an inquiry or reservation is created, both the customer and the admin should be notified.

Nonfunctional Requirements

They serve to explain the system's required standards so its performance in real life is excellent.

1. Anyone involved with the system, be it a customer or administrator, should be able to use it without difficulty.
2. Capacity: The system ought to be stable during times of increased use, new foods or new users.
3. Passwords should be saved using encryption such as through hashing. Protecting user data from outside interference and implementing web security rules is necessary for any website.
4. Time to Load: The application should be fast, responding immediately to requests, no matter if a user is in the frontend or backend area.
5. Availability: The system ought to be up and available most of the time. You should choose backup systems to avoid losing your data.
6. Anyone should be able to use the website with smooth results on popular browsers and on phones as well as computers.
7. The structure and documentation of the codebase should be good, so future developers can work on the code unhindered.
8. Privacy: The system should be designed so that it does not store unneeded sensitive information and keeps whatever it does store securely.

3.1.4 Context Diagram

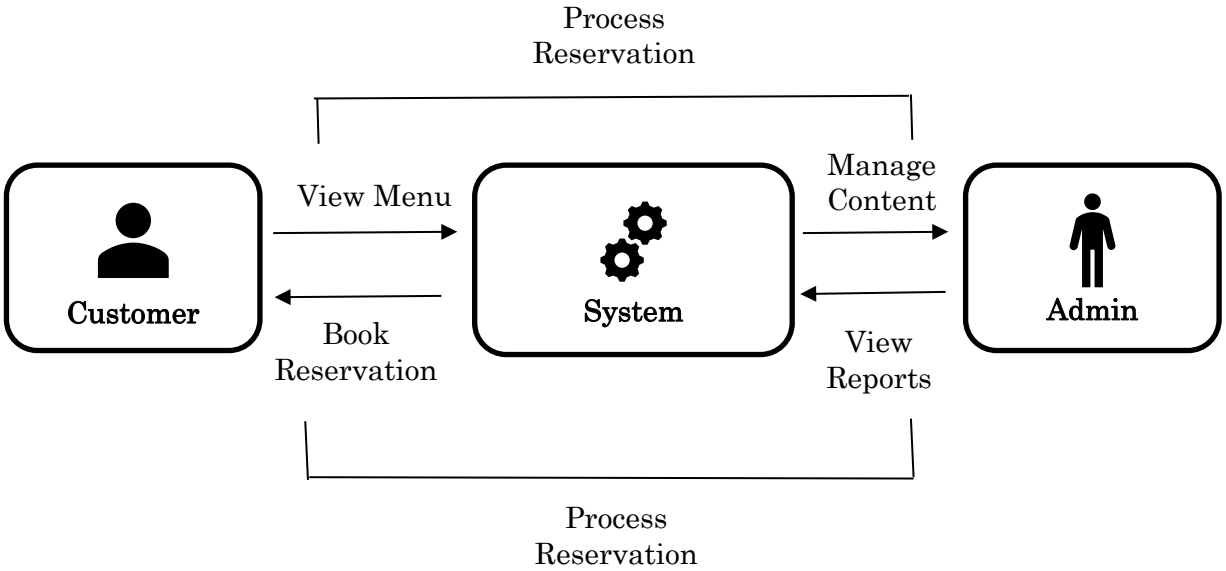
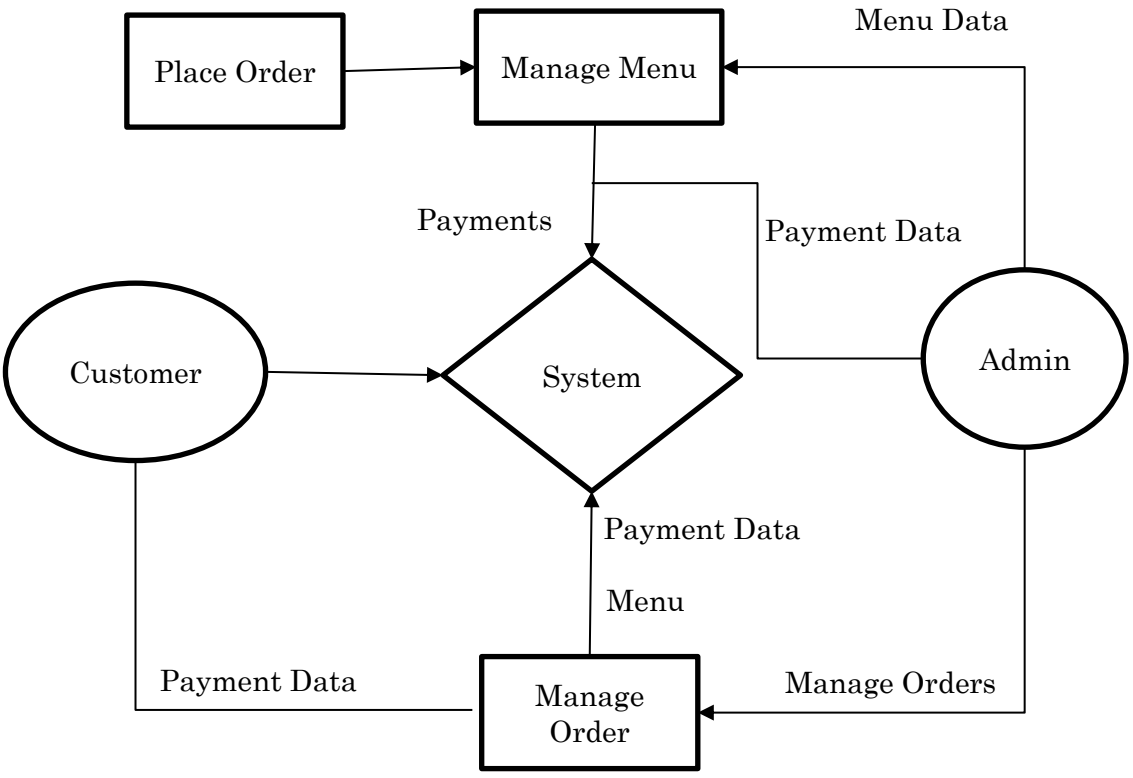


Figure 3.1.4: Context Diagram

3.1.5 Data Flow Diagram Level 1

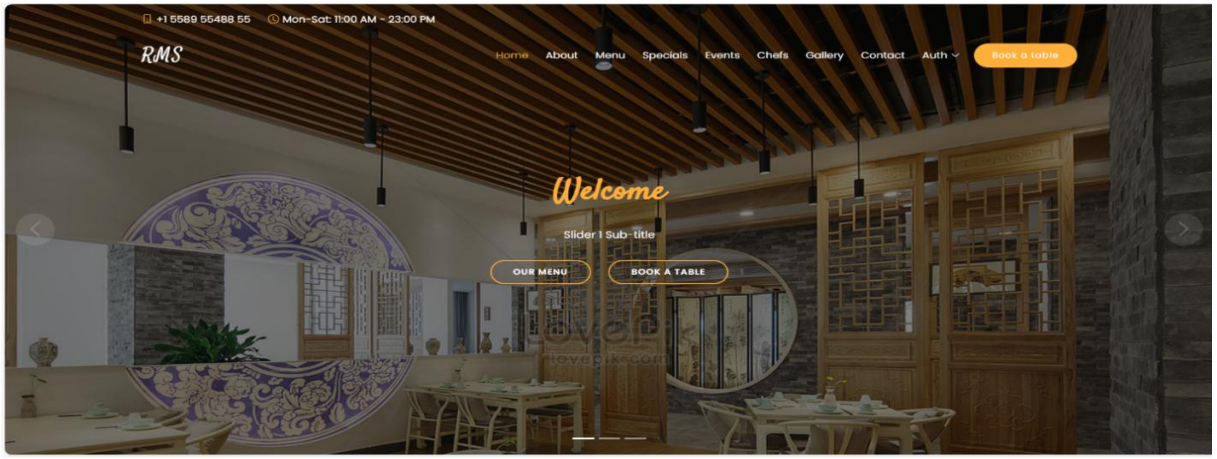
Figure 3.1.5: Data Flow Diagram



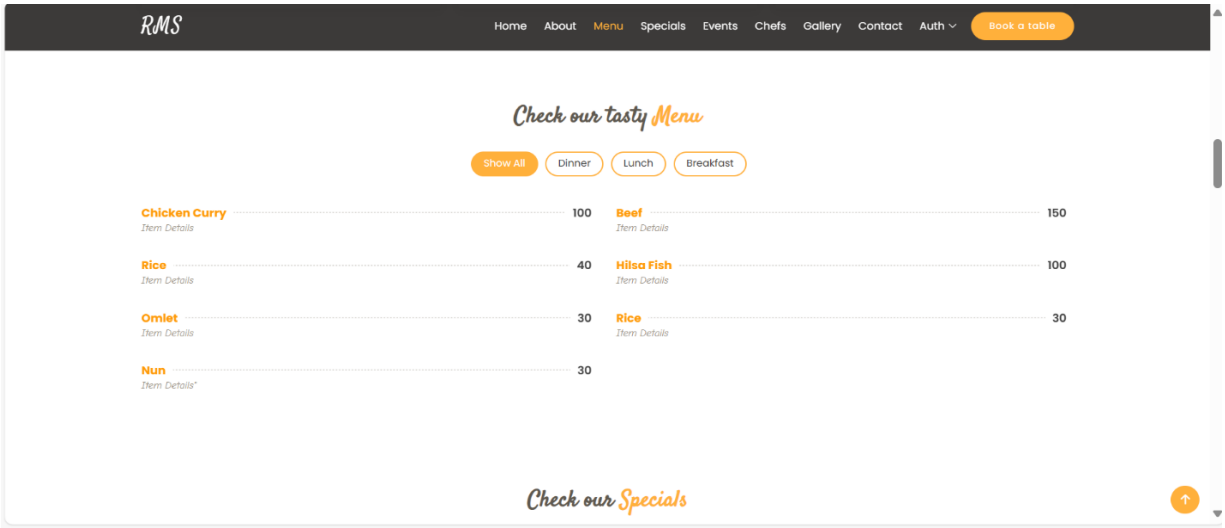
3.1.6 UI Design

Both customers and administrators in the restaurant industry find it simple and pleasant to work with the user interface of the system. The focus was on designing a plan that makes sense and ensures all required attributes for handling restaurant activities. This website was created using CSS, Bootstrap and Laravel in its frontend development process. It ensures the site is attractive and can be used easily from any device.

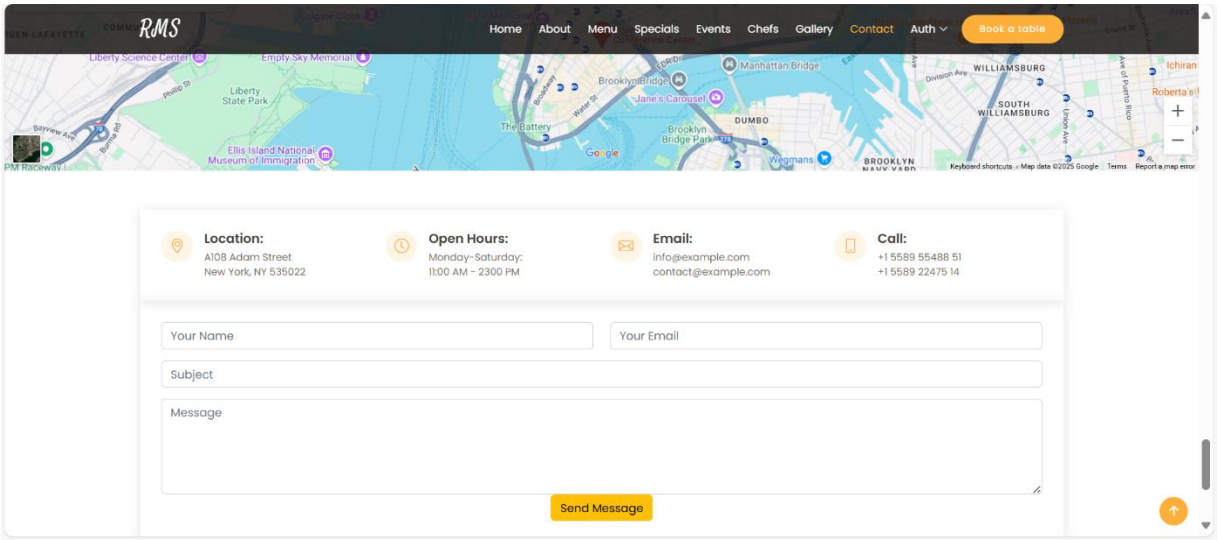
1. Homepage

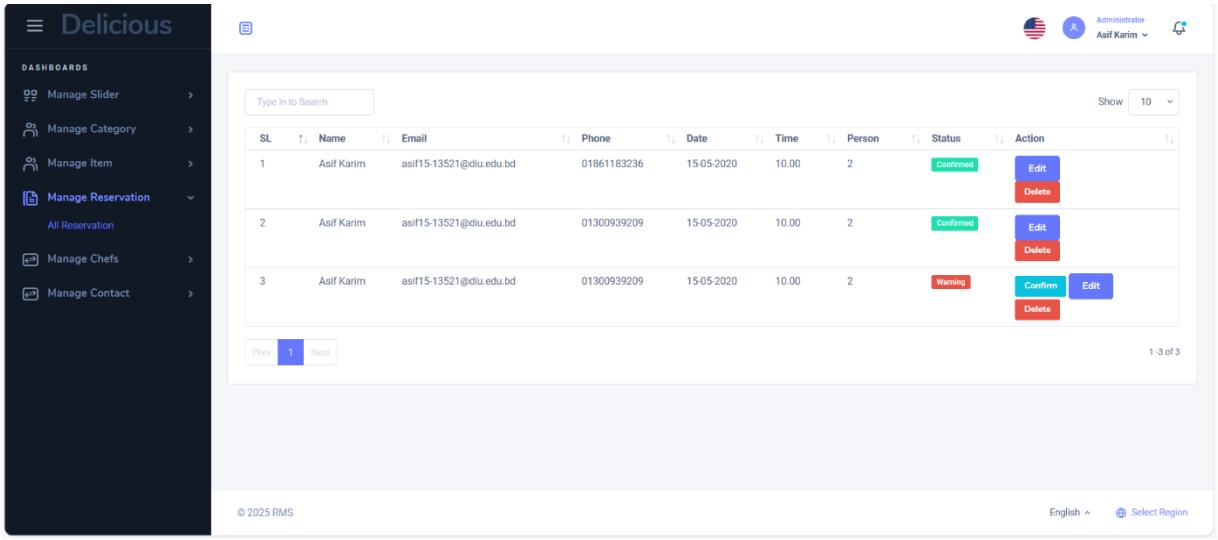


2. Menu

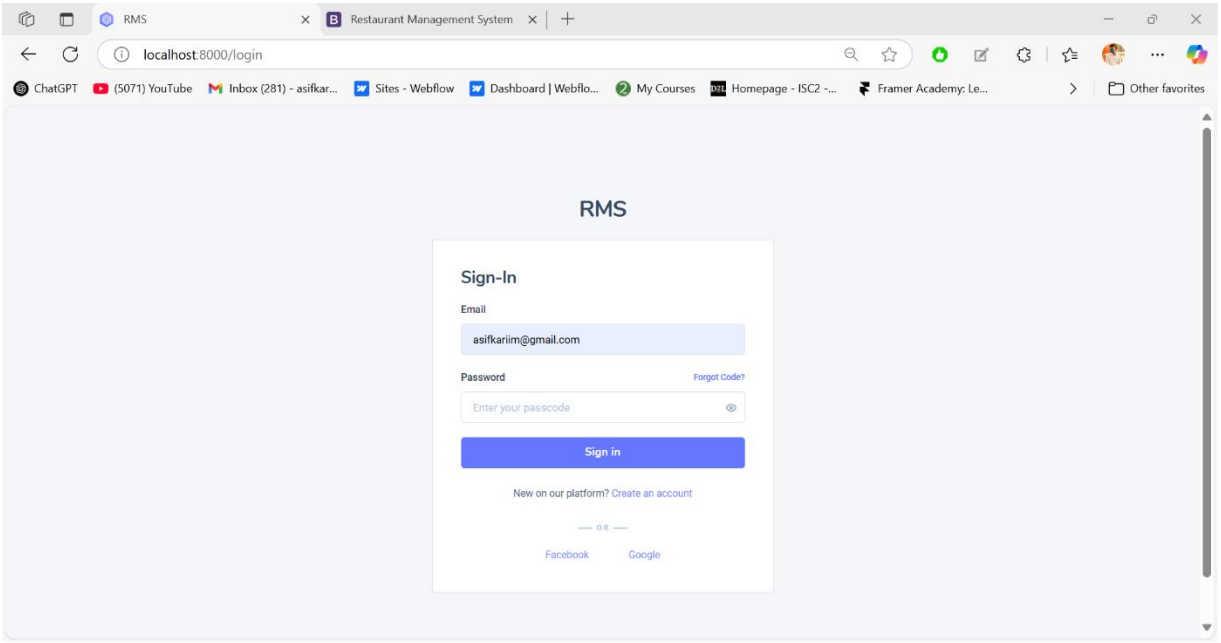


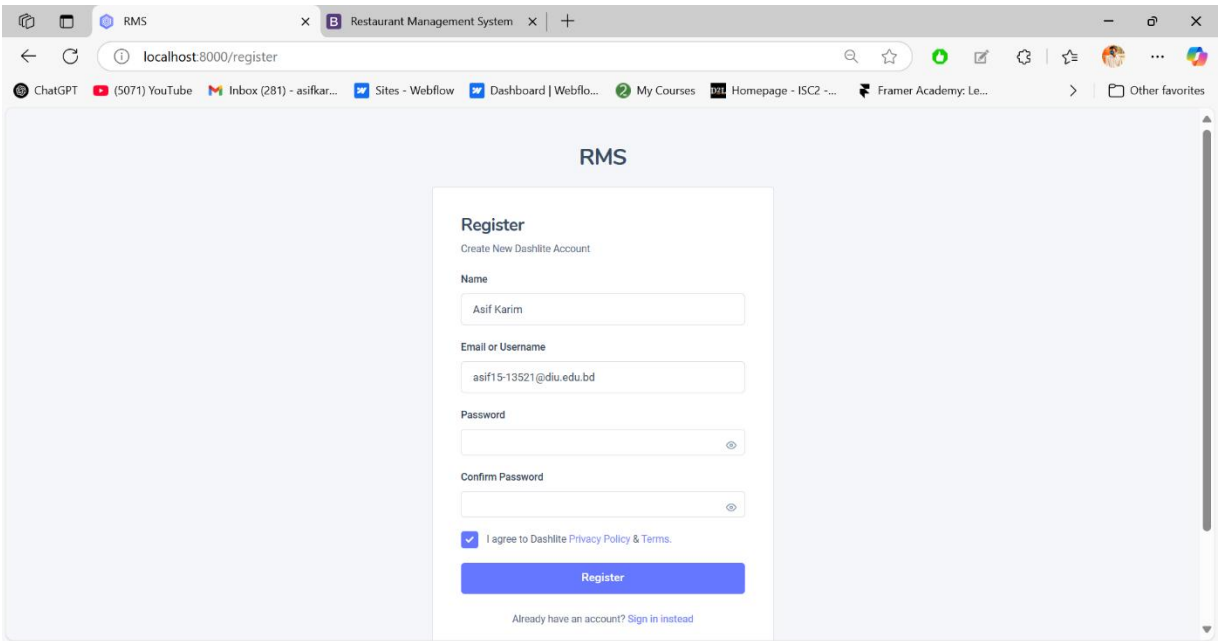
3. Reservation



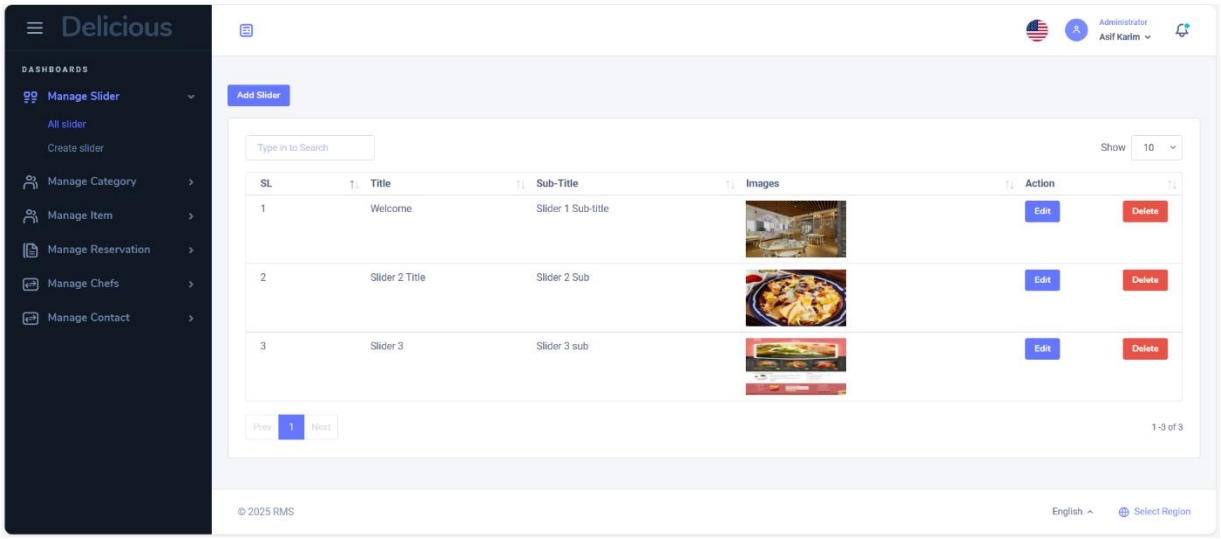


4. Log In and Registration





5. Admin Dashboard



3.2 Detailed Methodology and Design

The objective was to create a fully functional, easy-to-use, and secure system for restaurant operations, including menu management, reservations, chef profiles, and customer contact.

Alternate Solutions Considered

During the planning phase, we explored a few alternative technologies:

1. **WordPress with Custom Plugins**
   - *Pros*: Easy setup, plugin support for menus/reservations
   - *Cons*: Limited backend control, plugin dependency, less secure
   - *Reason for Rejection*: Not ideal for custom admin workflows and tightly integrated modules.
2. **Django (Python)**
   - *Pros*: Strong backend support, built-in admin interface
   - *Cons*: Less flexible for frontend, slower integration with custom UI
   - *Reason for Rejection*: Frontend styling and form control required more configuration than desired.

Selected Approach: Laravel with Blade

We selected Laravel for the backend framework and Blade for templating because it provided:

- A clean MVC architecture
- Secure routing and middleware support
- Built-in authentication and session handling
- Blade templating engine for fast, reusable frontend development
- Laravel's Eloquent ORM for smooth database interactions

This stack ensured rapid development, secure data handling, and ease of maintenance.

System Design Strategy

1. **MVC Pattern**: The system follows the Model-View-Controller pattern:
   - Models handle data and relationships (Reservation, Menu, Category, Chef, Contact)
   - Controllers contain business logic and manage data flow (AdminController, ReservationController, etc.)
   - Views are built using Blade to deliver a responsive and clean user interface
2. **Routing and Middleware**: Routes are declared in web.php, organized under authentication middleware to restrict admin access.

3. **Database Design**: Database tables are created using Laravel migrations:
    - **Tables**: users, reservations, menus, chefs, categories, contacts
    - Relationships are defined using Eloquent (e.g., Menu belongs to Category)
4. **Admin Panel**: A custom admin dashboard was built using Blade and Bootstrap. It allows content management without relying on external packages or JavaScript frameworks.
5. **Frontend Interface** The frontend is fully responsive, built using Blade and standard HTML/CSS (with Bootstrap or Tailwind depending on the setup), enabling users to browse menus, view chefs, book reservations, and submit queries.

Security and Validation

Laravel's built-in validation and CSRF protection were used across all forms. Authentication was applied using Laravel's default auth scaffolding, and input fields were validated at the controller level.

3.3 Project Plan

The development of the Restaurant Management System was carefully planned and divided into distinct phases to ensure timely delivery and smooth workflow.

Phase	Duration	Key Activities
Phase 1: Planning	Week 1	Requirement Gathering, competitor research, defining project scope
Phase 2: Design	Week 2	UI Mockups, Database design
Phase 3: Development 1	Week 3- 4	Implementing frontend using Blade, styling with CSS/Bootstrap
Phase 4: Development 2	Week 5-6	Backend development with Laravel controllers, models, and route logic
Phase 5: Integration	Week 7	Connecting frontend with backend, admin panel setup, reservation flow
Phase 6: Testing And Debugging	Week 8	Functional testing, bug fixes, final touches, and local deployment

Table 3.3: Project Timeline

3.4 Technology Stack

To ensure success, effective and latest technologies were used to develop the Restaurant Management System. All the tools were chosen to guarantee that the system could be expanded, protected and looked after with ease. They were able to make the development of the frontend and backend go smoothly because of the technology they used.

Backend

Technology	Purpose
PHP (Laravel)	Core Backend framework used to build the MVC architecture
Laravel Framework	Routing, controllers, authentication, middleware
MySQL	Relational Database used to manage structured data
Eloquent ORM	Laravel's built-in ORM to handle database relationships

Table 3.4.1: Backend Technologies

Frontend

Technology	Purpose
Blade Templates	Laravel's templating engine for dynamic HTML views
HTML5/CSS3	Base Structure and styling of all pages
Bootstrap	Responsive UI framework for layout and components

Table 3.4.2: frontend Technologies

Other Tools & Platforms

Tool	Purpose
XAMPP	Local development and server environment
phpMyAdmin	Database management interface
Git	Version control during development
Visual Studio Code	Code editor for writing and debugging

Table 3.4.3: Tools and Platforms

3.5 Summary

It highlighted the major decisions that influenced the design and development of the Restaurant Management System. Initially, we studied what the customer and admin needed to make sure the system would be practical. It was decided to look at various options for development such as WordPress and Django. Once the teams reviewed the benefits and drawbacks of both languages, Laravel was picked for being very flexible, safe and easy for programmers to use, mainly due to its clean MVC structure and the routing, security and data managing tools it features. It also pointed out how the chapter planned the stages of the project and the core programming technologies from Blade templates for the frontend to managing databases using Eloquent ORM from Laravel on the backend. Thanks to them, setting up the system was much clearer and more productive. We paid more attention to the tools involved in each part of the project, rather than simply listing the tasks. All in all, this chapter prepared the process for taking the idea and turning it into a working system, getting ready for what was done next.

Chapter 4

Implementation and Results

4.1 Environment Setup

Establishing the environment for development was essential when working on the Restaurant Management System. Using the correct combination of tools and platforms guaranteed an easy development and testing process. Below we will find a description of the basic tools needed for the process:

1. Local Development Environment

- **XAMPP:** Assists in creating a local server that operates with PHP and MySQL. It allowed me to easily test the application while I was developing it.
- **Composer:** An application that made it easier to get Laravel and maintain its dependencies.
- **Git**: Used to track the code changes being made and to help collaborate throughout the process of development.

2. Frontend Development

- We use Visual Studio Code (VS Code) to write code during the project. Users could support Laravel, Blade templates and Tailwind CSS and add extensions for improved productivity.
- Bootstrap and CSS helps to build responsive and trendy user interfaces in an easier way.

3. Backend Development

- **Laravel Framework**: Handled the main functionality by organizing routing, controllers, models and all migrations of the database.
- **PHP 8**: For business processing and operations on the web server.

4. Database

- **MySQL**: Used as a safe and well-backed database for saving menus, different categories, information about the chefs, reservations and messages sent by customers.

5. Deployment Preparation

- **AWS (Amazon Web Services)**: Opted for its ability to handle high traffic without issues and because it is reliable.
- **SSL Certificates**: Help ensure safety when data is transferred online.

4.2 Testing and Evaluation

Testing was very important during the process of creating the Restaurant Management System. It supported the development of an app that is safe, useful and effective for users in any situation. Various approaches to testing were used during the project to pay attention to every important detail of the system.

1. Unit Testing

To ensure they worked properly, each function, method and database connection was evaluated on its own. Tests were developed by using PHPUnit, one of the testing tools included in Laravel. Check that submitted and returned reservation information is stored and read accurately from the database.

2. Integration Testing: In this step, we checked that the parts of the system that worked together, for example frontend and backend, were functioning properly. Testing if a customer's information is recorded and handled as indicated by the system.

3. User Acceptance Testing (UAT): The system was tested by individuals who acted as end users which included restaurant teams and admins. Because of their feedback, the system was reliable and easy to use in real situations. Working on their ideas, adjustments were done to give customers a better experience.

4. Performance Testing: To see how the system acted when handling a lot of data, performance testing was done. The simulation included several users accessing the system all at the same time. Apache JMeter was used for testing response time, scalability and check stability when the system was at its highest usage.

5. Security Testing: During security testing, possible issues such as SQL injection, XSS attacks or people trying to gain access without permission were sought. OWASP ZAP was one of the resources used to verify that visitor data such as reservations and contact details, was kept secure.

.

4.3 Results and Discussion

Real-world uses allowed me to gather information on how reliable the application could be. In this part, we go over the results of various types of tests and highlight whether the system met its targets for performance, usability, functionality and security.

Functional Testing Results: All the important features such as booking, browsing the menu and handling content were assessed. Frontend and backend worked well together as data was able to move effortlessly across all parts. People could easily submit reservations, offer the necessary information and receive confirmations without any problems. Managing storeroom, food types, stock quantity and supplier information became relatively simple for admins using the dashboard. Although payment integration was not included this time, the system has been built to include it down the line.

Usability Testing Results: Results from usability testing show that during user acceptance testing, staff and managers found the system easy to learn and use. Most individuals found that they could begin using the dashboard almost right out of the box. According to testing results, it was recommended to offer new users tooltips and to improve the positioning of specific sections so that the most often used features would be easier to find. It was confirmed that the system was easy to use, yet could be improved through a few changes.

Performance Testing Results: During testing, the system was used by a large number of users at once to check its performance. The results confirmed that it continued to function smoothly and without any slowdowns. Even with increased load on the system, answers were delivered as expected, so it appears the backend is adjusted for growth.

Security Testing Results: The security test indicated that everything worked well and no major security weaknesses like SQL and cross-site scripting vulnerabilities could be found. The developer hashed passwords and also verified that form input is correct. A few issues were identified, including making session timeouts stricter and strengthening how passwords are stored.

Evaluation and Discussion: All in all, the Restaurant Management System delivered a system that is user-friendly, secured and functional for restaurant operations. It helps you with reservations, menu tasks and communicating with your customers. Due to user input, the application could add tracking of stock, a loyalty scheme for customers and the option to pay online.

4.4 Summary

In this chapter, I discussed using, testing and evaluating the Restaurant Management System. First, the project explained the settings for the development and LabVIEW IDE and then outlined the methods used to test the system. During testing, it was verified that the system functions well in managing the menu, making reservations and performing admin work. Tests proved that both the interface and the system's handling of traffic are excellent for all users. During security testing, the results were excellent and only minor suggestions for advancement were given. All things considered, the system works well and offers effective ways to manage a restaurant. The chapter showed that it has already met its major aims, but there are still areas that can be improved by updating the system.

Chapter 5

Engineering Standards and Design Challenges

5.1 Compliance with the Standards

The team worked hard to ensure the system followed the best standards and established practices in software development. While working on the project, special efforts were made to maintain the system's security, easy maintenance and usability.

Here are the key areas where the system aligns with industry standards:

- **Security Standards:** Using Laravel, the system ensures that the admin panel can only be accessed by those with permission. All passwords are protected with security encryption using hashes and input checks are used to defend from threats such as SQL injection.
- **Coding Standards:** Codes have been written using the best coding techniques for PHP and Laravel development. It contains sections (modules) that make it simple to understand and extend. As a result, the system will be easier to maintain and grow as time goes on. The system is accessed easily on all devices because it follows responsive web design
- **Accessibility Standards:** Accessibility was considered, so the interface uses large enough fonts, allows for using the keyboard instead of the mouse and employs UI elements that use enough contrast.
- **Data Protection:** The site uses data protection principles when handling the personal information of its users. This method of access control makes sure that unauthorized users cannot access private regions. The system also specifies how data should be gathered, stored and handled.

This way, the system becomes efficient, simple to use, dependable, secure and suitable for use.

.

5.1.1 Software Standards

To guarantee the system was and continues to be reliable, secure and easy to maintain, professional standards and practices were used during its development. Here is an overview of the key standards we used when developing the platform:

- **Coding Conventions:** All the code was written using the PSRC-1, PSR-2 and PSR-4 standards from the PHP-FIG. By doing this, it is simpler and clearer for other developers to work with and contribute to the project.
- **Framework Best Practices:** When creating the system, it follows the Model-View-Controller approach and other guidelines recommended by Laravel. The approach makes it possible to maintain the app easily and also update it in the future.
- **Version Control:** Our team used Git for version control so we could spot all project changes, help other team members and review all the past changes. All the code from the project is kept in GitHub, so collaboration is simple and everyone can have access to it.
- **Database Design:** Database Design: By applying relational normalization, we efficiently organize our data. With the combination of MySQL and Laravel's Eloquent ORM, we organize our data so that it remains uniform and does not repeat itself.
- **Security Standards:** Security was given top priority during the entire development process. We adopted secure coding to prevent attacks such as SQL injection, XSS and CSRF. The system was improved using the query builder and CSRF protection that are available in Laravel.
- **Documentation:** Every project is accompanied by useful and clear documents, including any necessary comments in the code and setup steps. Thanks to this, both developers, system administrators and users can operate the system readily.

5.1.2 Hardware Standards

The hardware requirements for the Restaurant Management System have been carefully set to ensure smooth performance and reliability. These standards help support the system's features and allow it to scale as needed:

1. **Server Requirements:** The system may be installed on-premise or in the cloud, according to how the restaurant decides. Our team advises that the best way to use these servers is with the following specifications:
    - **CPU:** Must have a dual-core processor of 2.5 GHz or more, for example an Intel Core i5.
    - **RAM:** 4 GB of RAM is bare minimum, though 8 GB is suggested to run the app for a larger number of friends.
    - **Storage:** At least 50 GB of free space on an SSD is required and you can upgrade as your data increases.
    - **Network:** set up on the cloud, make sure internet is consistent and has a speed of at least 10 Mbps.
2. **Client Hardware:** Different User Interfaces are built to function well regardless of what device is being used. It is recommended to look for computers with these specs:
    - **Desktops/Laptops:** should have at least 4 GB RAM and a 1.8 GHz processor and the machine should be running Windows, macOS or Linux.
    - **Mobile Devices:** can use smartphones or tablets with 2 GB RAM or more that have Android 6.0 or the 10.0 version of iOS or higher. Products with larger memory (for instance, 4 GB) run faster.
3. **Peripheral Devices:** To help restaurant employees with ordering, running the POS and inventory, various peripheral devices are often used in combination with the system. Recommended hardware includes:
    - **Barcode Scanners:** USB or Bluetooth scanners to read all types of standard barcodes.
    - **Receipt Printers:** Receipt printers with a speed of at least 150 mm/second that can link with your POS system.
    - **Touch-Screen Monitors:** Screens measuring 15 inches or more, with a minimum resolution of 1024x768, help in easy and efficient POS transactions.

4. **Backup Systems:** Frequently saving copies of your data will prevent you from losing it. The backup configuration should involve the following:
    - **External Storage:** Should have a local external storage of at least 1 TB for your backups.
    - **Cloud Storage:** Using AWS or Google Cloud enables cloud storage for backups to be used in case of disaster..

5.1.3 Communication Standards

Effective communication in the restaurant is made possible through the use of key guidelines and data styles in the system. For safe communication between the client and server, HTTP is used and for real-time updates, WebSockets are applied. The data used by Parse is exchanged in JSON to make the process smooth and fast. Messages through push notifications, SMS and emails keep customers and employees up-to-date on their orders and any promotions. As there may be many customers wishing to pay at once, message queues like RabbitMQ are used to ensure the site stays responsive. Verifying personal information is safe as communications are secured with TLS and checks made by two-factor authentication limit entry to admin and staff accounts.

5.2 Impact on Society, Environment and Sustainability

With the Restaurant Management System streamlining many tasks, more jobs in technology and data management can be created. It helps to protect the environment as people use electronic reservations, orders and menus instead of paper. Moreover, managing inventory more properly allows for a reduction in food waste. Supporting green choices and environmentally friendly routines in the restaurant industry helps contribute to its growth.

5.2.1 Impact on Life

Operating restaurants is less complicated thanks to the Restaurant Management System. Reservations, checking out menus and interacting with the restaurant online become easiest when they are available on apps. It makes the service faster for staff and reduces waiting time for customers. Being able to control table reservations, menu items and opinions from guests gives managers of restaurants more satisfaction and makes operating smoother. On the whole, this approach meets the expectations of today's society which favors digital convenience when handling everyday activities.

5.2.2 Impact on Society & Environment

The adoption of the Restaurant Management System is good for both the environment and society. When restaurants use the system, it reduces the use of paper which is better for the environment. It also helps the staff oversee inventory better, meaning they can reduce any unnecessary waste of foods. Through its technology, the system ensures restaurant services are easier for people with disabilities to access. Furthermore, it helps create new jobs in technology and services which benefits the local economy. Because restaurants can look at customer data, they can ensure that each visitor enjoys their visit. To sum up, the system supports environmental actions and adds value to services given to people living in the area.

5.2.3 Ethical Aspects

In both making and using the Restaurant Management System, care should be given to ensure fairness, privacy and transparency are maintained. The main concern is ensuring customer data is protected by strictly complying with GDPR or similar regulations. As a result, credit card details, individual choices and contact information are protected. The designers should ensure that there is no bias in customer recommendations and promotions by setting guidelines for such activities. One should also take care not to endanger people facing challenges with technology or those likely to lose their jobs to automation. People need to understand clearly what data is being collected and exactly how it will be put to use. Furthermore, the system should have processes that do not contribute to issues such as more waste or harm to the environment. This ensures each part of the system functions ethically and does not harm anyone.

5.2.4 Sustainability Plan

The strategy addresses reducing the Restaurant Management System's adverse impact on nature, bolstering continued growth and maintaining its effectiveness. Moving the system to cloud infrastructure will reduce the need for servers and help save power. It will further use storage and processing methods that use less energy and can grow with the number of users. So that it can grow with time, the design will ensure that new modules can be easily implemented as restaurant technology develops. It allows the system to be updated easily and frequently and significant changes are not required. Updating the software frequently will allow the system to stay up-to-date and meet users' changing requirements. Moreover, the restaurant industry is made greener by helping businesses keep their food waste down and choose eco-friendly suppliers. Ensuring that the system is efficient in how it works and also protects the environment helps a business flourish and use resources responsibly.

5.3 Project Management and Financial Analysis

Budget Analysis

The budget for the Restaurant Management System must cover the costs of the first phase and also the costs for running it in the future. Here, I will cover in detail the common costs of a road trip.:

1. **Development Costs:**
    - **Software Development:** Need to pay the developers, designers and project managers who are creating the system. Programming, testing and building the components of the front and back end are the central tasks.
    - **Frontend and Backend Development:** Designing the user interface and improving the user interface are examples of frontend development. It is the backends' job to oversee database operations, run all the system logic and connect with services from other providers.
    - **Quality Assurance (QA):** Testing is necessary to ensure the system can be trusted. Money invested in testing makes it clear that the application functions properly on many devices.
2. **Deployment and Maintenance Costs:**
    - **Cloud Hosting and Infrastructure:** Because AWS, Google Cloud and Azure host the system, you will pay for storage, usage of computer resources and bandwidth regularly. That is why the network maintains scalability, reliability and accessibility.
    - **Maintenance:** After the application is published, it must be regularly checked for bugs and improved by implementing suggestions from users.
    - **Marketing and Promotion:** Distributing budget to support social media, online ads, SEO and writing content to encourage people to try and stay on the platform.
3. **Operational Costs:**
    - **Customer Service and Support:** These expenses are used to respond to customers and find solutions for any problems they have.
    - **Legal and Licensing Fees:** Fees for registering and maintaining licenses, preparing rules and policies needed by the law and complying with all applicable regulations.

Alternate Budget Proposal

If you wish to oversee your budget while dealing with uncertain changes in the market, a flexible plan can support the business without falling below quality::

1. **Outsourcing Development:** Graphic design, UI/UX and other specific development tasks can be outsourced to freelancers or workers in other countries. When this is done in regions where wages are lower, high standards can be reached at a lower labor cost.
2. **Phased Development:** Working in steps, the team can build the project, starting with an MVP. Key features like order management, processing payments and some reporting would be included in the MVP. At the beginning, you can use only the basic features, but when you require additional tools such as advanced analytics or feedback collection, both revenue and spending can be controlled.
3. **Marketing Budget Adjustment:** You could begin by reaching out to the community, collaborating with other organizations and having supporters spread the word. After popularity increases, it makes sense to focus some funds on paid advertising efforts.
4. **Cloud Service Optimization:** Identifying the right cloud services and adding or removing resources as needed saves money on hosting costs. It is more economical to begin on a basic cloud plan and increase as your company grows.

Revenue Model

Exploring a number of revenue sources can ensure that the system remains sustainable and makes a profit.:

1. **Subscription-Based Model:** For this business, a monthly or annual subscription from restaurants could be the primary income source. The price may change depending on the level of users, restaurant space or which features the subscription provides. Using this strategy allows you to have a regular and predictable income..
2. **Freemium Model:** Here, individuals can use the basic feature set for free, while more elaborate features must be paid for. Users are easily attracted to freemium models and later encouraged to choose paid plans.
3. **Transaction-Based Model:** One way to earn is by charging users a small amount for every transaction made on the technology. Your service could include order handling fees, costs for payment gateways and a sales commission, growing the business's revenue in line with the number of orders.
4. **One-Time Setup Fee:** As well as subscriptions, the system might require a first-time fee set up when you start using it. The fee is meant to cover setup and training of staff, customize the system with a company logo and connect the operation's POS and inventory tools.

5. **Advertising Revenue:** The platform may earn extra profits by including advertisements from food suppliers, shops with kitchen items and delivery partners. Even though this kind of platform needs many users to turn a profit, it may still provide a steady stream of extra income with other revenue sources.

5.4 Complex Engineering Problem

5.4.1 Complex Problem Solving

In this section, the system's problem-solving aspects will be mapped to different categories of complex problem-solving. Each mapping will explore the rationale behind the decisions made at each stage, considering the requirements of the project. The following table will provide an overview of how each problem-solving category aligns with the development process.

EP1 Dept of Knowledge	EP2 Range Of Conflicting Requirements	EP3 Depth of Analysis	EP4 Familiarity of Issues	EP5 Extent of ApplicableCodes	EP6 Extent Of Stake-holder Involvement	EP7 Interdependence
✓	✓	✓				✓

Table 5.4.1: Mapping with complex problem solving.

K3 Engineering Fundamentals	K4 Specialist Knowledge	K5 Engineering Design	K6 Engineering Practice	K8 Research Literature
✓	✓	✓	✓	

Table 5.4.1.2: Mapping with knowledge Profile.

Justification for EP Mapping:

1. **EP1 Depth of Knowledge:** The system required knowledge of MVC architecture, Laravel's routing and controller logic, Blade templating, database relationships with Eloquent ORM, and secure authentication. Developing a complete backend-admin and frontend-customer flow demanded the integration of multiple web technologies cohesively.
2. **EP2 Conflicting Requirements:** The platform needed to offer an intuitive and easy-to-use interface for customers while providing in-depth controls for administrators. Balancing simplicity, validation strictness, performance, and security required trade-offs during design and implementation.
3. **EP3 Depth of Analysis:** Reservation logic, chef management, and category filtering

all required specific conditions and validation logic. Building these modules demanded careful data planning, relationship handling, and scenario testing.

4. **EP7 Interdependence:** The system's modules menu, reservations, chefs, and contact are interlinked. A modification in one module (e.g., removing a category) affects menu visibility and potentially reservation inputs. Each component depends on the others to work as a unified whole.

Justification for Knowledge Profile Mapping:

1. **K3 Engineering Fundamentals:** Required understanding of basic web development principles including HTTP, form submission, session management, database structuring, and control flow.
2. **K4 Specialist Knowledge:** Applied Laravel-specific features such as controllers, Blade views, middleware, migrations, and Eloquent ORM to build a secure and interactive system.
3. **K5 Engineering Design:** The UI was designed with a clear focus on user experience, while the backend was structured for scalability. The admin dashboard and frontend views were customized to support modular growth and user control.

4. **K6 Engineering Practice:** Development followed SDLC stages including requirement gathering, design, implementation, and testing. Real-world tools such as Git (for version control), XAMPP (local server), and structured folder systems were used. Form validations, middleware protection, and modular controller design demonstrate adherence to industry practices.

5.4.2 Engineering Activities

EA1 Range of resources	EA2 Level of Interaction	EA3 Innovation	EA4 Consequences for society and environment	EA5 Familiarity
✓	✓	✓		

Table 5.4.2: Mapping with complex engineering activities.

Justification for Engineering Activities Mapping

1. **EA1 Range of Resources:** The project utilized Laravel, PHP, MySQL, Blade templates, CSS, Bootstrap, Git, XAMPP, phpMyAdmin, and VS Code—covering a broad range of development and deployment tools.
2. **EA2 Level of Interaction:** The system manages interactions at multiple levels: customers use the public site, admins use a backend dashboard, and various backend modules communicate with each other. For example, deleting a menu category affects how the menu page displays items. Each module is built to respond to data from others, ensuring consistency across the platform.
3. **EA3 Innovation:** The admin dashboard was developed from the ground up, customized to the needs of restaurant owners. Unlike pre-built CMS platforms, the system provides full control over frontend content and real-time updates through secure routes and forms.

5.5 Summary

The chapter discussed applying engineering standards, the project's effects on society, the environment and sustainability, the management of the project, its financial aspect and problem-solving. Engineering tasks were grouped according to their problem-solving function and connected to different engineering practices, to show how each phase of the project was analyzed and checked by all stakeholders. Following the recommended standards resulted in a dependable and well-built big data system and ethics and sustainability were considered too.

Chapter 6
Conclusion

6.1 Summary

The project aimed to build a system capable of addressing the mentioned challenges. Because the project was developed using a careful strategy and detailed plans, all its requirements at the start were properly fulfilled. Developers used widely accepted standards in software engineering to ensure that everything performed smoothly and without problems. The system went through testing and evaluation and proved it can be used in practice. Furthermore, the project took care to consider social, environmental and ethical issues and followed responsible growth principles. A thorough budget and viable plan for generating income were major components of the project's finances. To conclude, the project was carried out successfully as planned. Even after the system is fully running, there is always a chance to make it bigger, add fresh features and enhance the way people use it while new technologies become available.

6.2 Limitation

Though the system accomplishes its top objectives, there are still some places where improvements could be made in the future. Since there was not much time and resources, we lacked room to teach features and integrations that came after the essential ones. Handling a large number of users can weaken the system's performance. The user interface is working, but it could be improved to make using the app smoother and more enjoyable. Additionally, some technologies used by the system may not always work with new game consoles, so developers must continuously update it to keep up. Furthermore, because the system has been tested in several practical situations, it is necessary to continue further testing and ask for more feedback from users to find unusual problems and strengthen it for various situations.

6.3 Future Work

Even though the current approach works well, there are plenty of chances to improve in the future. Improving the ability of the system to handle a large number of users is the main concern for future development. With a larger user base, the system needs to be updated with new infrastructure or use better strategies to ensure it runs smoothly. Applying cloud technology or distributing the architecture could be necessary for achieving scalability. Adding intelligent algorithms for personal recommendations, letting AI assist in customer service and improving how products are filtered are good ideas too. This would help YouTube enhance its personalized features and make them more fitting for different users. Better user experience can result from making the design and interface of the system more effective. Testing the website with users and collecting their feedback allows you to improve its ease of use. Being more responsive and adapting to mobile platforms will allow more users to access your website. In addition, restaurants can enhance their operations and improve the system by adding services for payments, analytics and deliveries. As a result, the system would respond well to changes made by different businesses. Security needs to be monitored all the time. We should consider using enhanced encryption, adding a multi-factor authentication system and keeping an eye out for potential threats. Overall, even though the system supports its goals as planned, additions in scalability, features, design, integration and security would assure it maintains its edge and effectiveness with the rapid technological progress.

References

[1] Laravel Documentation, "Laravel Documentation (*Laravel.com*, [Online]. Available: https://laravel.com/docs/11.x

[2] Tailwind Labs, "Tailwind CSS Documentation," *Tailwindcss.com*, 2024. [Online]. Available: https://tailwindcss.com/docs/installation

[3] Vite Team, "Vite Documentation," *Vitejs.dev*, 2024. [Online]. Available: https://vitejs.dev/guide/

[4] PHP Group, "PHP Manual," *PHP.net*, 2024. [Online]. Available: https://www.php.net/manual/en/

[5] Oracle Corporation, "MySQL Reference Manual," *MySQL.com*, 2024. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/

[6] M. Stauffer, *Laravel: Up & Running*, 3rd ed., Sebastopol, CA: O'Reilly Media, 2019.

[7] C. Northwood, *The Laravel Application Development Cookbook*, Birmingham, UK: Packt Publishing, 2023.

[8] Open Web Application Security Project (OWASP), "OWASP Top Ten," *OWASP.org*, 2021. [Online]. Available: https://owasp.org/www-project-top-ten/

[9] Bootstrap Team, "Bootstrap Documentation v5.3," *getbootstrap.com*, 2024. [Online]. Available: https://getbootstrap.com/docs/5.3

[10] Visual Studio Code, "VS Code Documentation," *code.visualstudio.com*, 2024. [Online]. Available: https://code.visualstudio.com/docs

[11] Git SCM, "Pro Git Book," *git-scm.com*, 2023. [Online]. Available: https://git-scm.com/book/en/v2

[12] DigitalOcean, "How to Use Laravel Eloquent ORM," *DigitalOcean Community*, 2023. [Online]. Available: https://www.digitalocean.com/community/tutorials

[13] Laravel News, "Understanding Laravel Middleware," *laravel-news.com*, 2023. [Online]. Available: https://laravel-news.com/understanding-middleware

[14] Figma, "Figma Design Platform," *figma.com*, 2024. [Online]. Available: https://www.figma.com

[15] Laracasts, "Laravel from Scratch Series," *laracasts.com*, 2024. [Online]. Available: https://laracasts.com/series/laravel-8-from-scratch

[16] MDN Web Docs, "HTML Forms Guide," *developer.mozilla.org*, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Forms

[17] W3C, "Web Content Accessibility Guidelines (WCAG) 2.1," *w3.org*, 2023. [Online]. Available: https://www.w3.org/WAI/standards-guidelines/wcag/

[18] XAMPP Apache Friends, "XAMPP for Windows," *apachefriends.org*, 2024. [Online]. Available: https://www.apachefriends.org/index.html

[19] Laravel Forge, "Server Deployment Made Simple," *forge.laravel.com*, 2024. [Online]. Available: https://forge.laravel.com

[20] FreeCodeCamp, "PHP Beginner's Handbook," *freecodecamp.org*, 2024. [Online]. Available: https://www.freecodecamp.org/news/php-tutorial/

193-15-13521

ORIGINALITY REPORT

12%	9%	2%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	7%
2	Submitted to United International University Student Paper	1%
3	Submitted to University of Ulster Student Paper	<1%
4	dspace.daffodilvarsity.edu.bd:8080 Internet Source	<1%
5	Submitted to University of Kansas Student Paper	<1%
6	Submitted to Universiti Teknologi MARA Student Paper	<1%
7	Submitted to Turku University of Applied Sciences Student Paper	<1%
8	arxiv.org Internet Source	<1%
9	Submitted to University of Northampton Student Paper	<1%
10	internship.daffodilvarsity.edu.bd Internet Source	<1%
11	Submitted to University of Finance – Marketing Student Paper	<1%
12	media.neliti.com Internet Source	<1%