VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
INFORMATION TECHNOLOGIES STUDY PROGRAM

Courseworks

Web application development

Done by:
Oleksandr Rotaienko

Supervisor:
Dr. Joana Katina

Vilnius
2024

Contents

1 Introduction

1.1 Topic Selection

I chose the topic of website development because it is essential in today's business environment. As a chef in a restaurant that needed a website for the menu and a member of the organization "UYGL" which also needs a website, I have firsthand experience of the requirements and benefits of having a well-designed website.

1.2 Relevance

In the modern world, nearly every company integrates websites and web development into their operations. It is difficult to find a business that does not have its own website. Moreover, many restaurants, even those without a complete website, often have an online or electronic menu. This highlights the growing importance of digital solutions in the restaurant industry, where efficiency and customer experience are greatly enhanced by effective web tools.

Increasingly, customers prefer the convenience of making orders directly from their smartphones without needing assistance from a waiter. This self-service option not only saves time but also provides customers with greater control over their orders, allowing them to browse the menu, select items, and customize their choices at their own pace. By scanning a QR code placed on the table, patrons can access the menu, place orders, and even complete payments digitally. This shift towards mobile ordering reflects a broader trend in the hospitality industry, where enhancing customer autonomy and streamlining the ordering process are key to improving service quality and overall customer satisfaction.

1.3 Benefits of the Developed System

The developed system will allow for comprehensive editing and management of the restaurant menu. Administrators can add new dishes, create categories, rearrange dish types, and manage toppings with great flexibility. I have also implemented a feature that allows the administrator to add waitstaff, providing them with an interface for taking orders. Additionally, I created an interface for restaurant guests, accessible via a QR code placed on the table, where they can pay for their orders, split the bill (pay for individual items), and even order more items directly to their table. The system is designed to be intuitive and easy to use, even for those without IT expertise, and supports mobile devices to ensure accessibility for users on the go.

1.4 Goal of the Work

The goal is to create a user-friendly website with an admin interface that simplifies menu management for restaurant staff, supports the addition and management of waitstaff, and provides a seamless ordering experience for restaurant guests. The system also supports mobile access to ensure that users can interact with it from their smartphones and tablets.

1.5 Tasks of the Work

1. Design a responsive website layout.

2. Develop a detailed and intuitive admin interface for managing the menu and staff.
3. Ensure mobile compatibility for all user interfaces.
4. Implement features for adding, editing, and organizing menu items, as well as managing add-ons.
5. Develop an interface for waitstaff to efficiently take and manage orders.
6. Create a guest interface accessible via QR code for ordering, bill payment, and splitting payments.

1.6 Expected Results

The expected outcome is a fully functional website that allows restaurant staff to easily manage the menu and waitstaff, enhancing customer experience and operational efficiency. Additionally, the system will provide guests with an easy-to-use interface to manage their dining experience, from ordering to payment, ensuring a modern and convenient service that meets the demands of today's digital-savvy customers.

2 Analysis of Similar Systems

2.1 Information Gathering on Similar Systems

To understand the landscape of restaurant menu management systems, I researched several websites and platforms that offer these services. Unfortunately, I couldn't access the admin panels of these systems directly. Instead, I focused on analyzing the admin panel and functionality of the restaurant company WAKE WAY, with which I am familiar. Additionally, I analyzed the GrillLondon website for its user-ordering process.

2.2 Analysis of WAKE WAY's Admin Panel

After analyzing the admin panel of WAKE WAY, I can provide the following insights:

Disadvantages

1. **Limited Menu Customization**: The system does not allow for adding or changing types of dishes. This limitation extends to the inability to rearrange the order of dish types.
2. **Topping Management**: There is no functionality to add toppings to dishes, nor can prices be assigned to these toppings from the admin panel.
3. **Icon Management**: While the system does allow selecting icons to display next to dishes (such as vegan or spicy icons), these icons cannot be added, deleted, or modified through the admin interface.

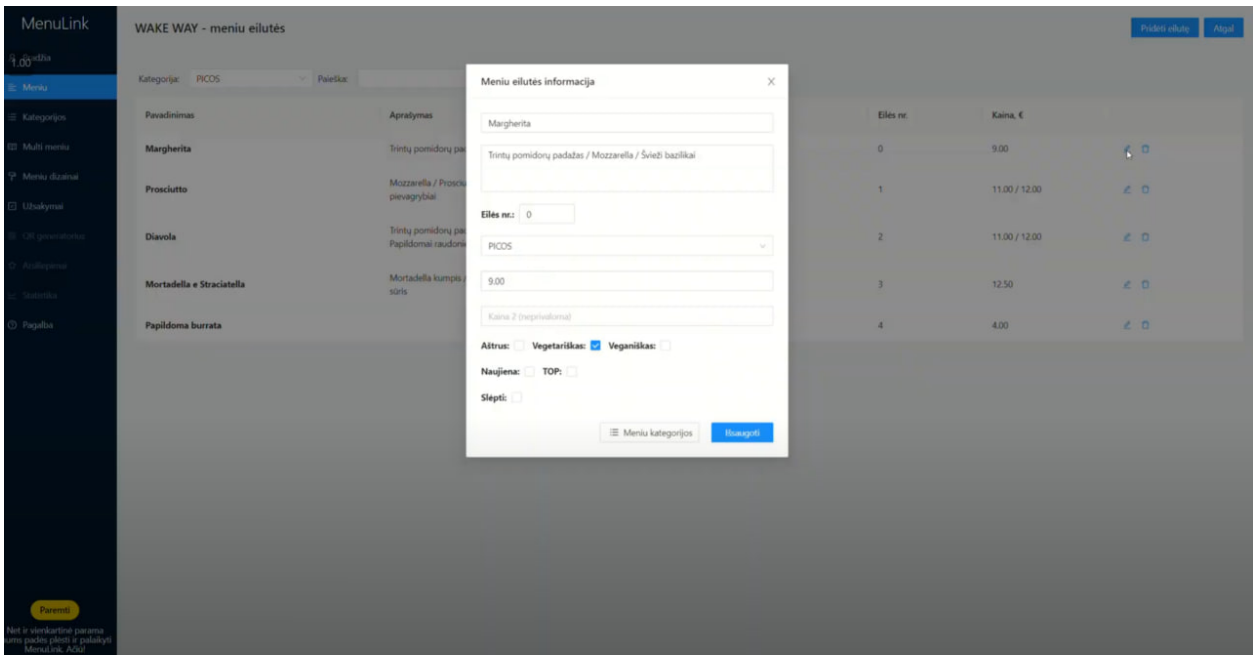

Figure 1. Wake way menu

Advantages

1. **Predefined Icons**: The ability to choose predefined icons for dishes is a helpful feature. These icons can visually indicate specific dish attributes, such as being vegan or spicy, although the set of icons is fixed and cannot be edited from the admin panel.

2. **User-Friendly Interface**: The interface is quite straightforward and easy to understand, which makes it accessible for users who may not have an IT background.

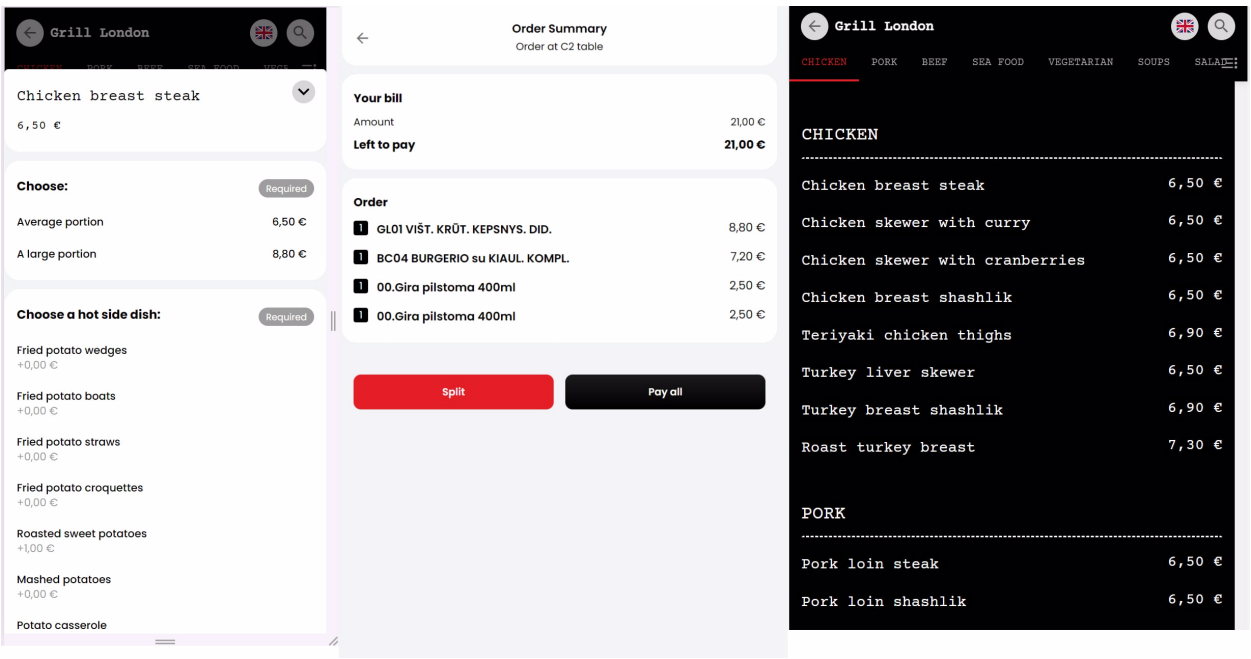

Figure 2. GrillLondon

2.3 Analysis of GrillLondon's Website

Advantages:

1. **Minimalist design:** The GrillLondon website uses a very minimalist approach to design, which positively impacts its clarity and ease of use. The clear structure and ample spacing between elements make the site intuitive and easy to navigate, even for first-time users.

2. **Online payment option:** Users can pay for their orders online, which adds convenience and speeds up the service process. This also reduces the burden on waitstaff and allows customers to handle payment on their own.

3. **Bill-splitting feature:** The website offers the ability to split the bill, allowing customers to pay for individual items. This is particularly useful for group orders where each customer wants to pay for their own part.

4. **Access to menu and add-ons:** Users can view the menu and additional options (add-ons) for each dish, enabling them to better tailor their orders to their preferences.

Disadvantages:

1. **Lack of dish photos:** The GrillLondon menu lacks photos of the dishes, which is a significant drawback. Photos would help users better understand what they are ordering and make more informed decisions.

2. **No option to place orders:** Despite the convenient design and features, the site does not allow customers to place orders themselves. This could be a critical drawback, as many users expect to be able to order directly through the website.

2.3.1 Determining Functional Requirements

Based on my analysis of the WAKE WAY admin panel and the research conducted on other similar systems, I outlined the essential functional requirements for an effective restaurant menu management system. These requirements include:

1. **Menu Customization:** The system should allow administrators to add, edit, and delete dishes, categories, and toppings. It should also support the reordering of menu items to match the restaurant's current offerings and priorities.

2. **Intuitive Interface:** The admin panel should be designed to be user-friendly, even for staff members without advanced technical skills. This includes clear navigation, large buttons, and minimal steps for completing tasks.

3. **Integration with POS Systems:** Seamless integration with point-of-sale (POS) systems is essential to streamline operations, synchronize orders, and manage payments efficiently.

4. **Mobile Compatibility:** Given the increasing reliance on mobile devices in restaurant operations, the system should be fully responsive, allowing staff to manage the menu and orders from smartphones and tablets.

5. **Order Management:** The system should include features that allow waitstaff to create, and manage customer orders. Additionally, customers should be able to place orders themselves via a mobile interface, with options for bill splitting and online payments.

6. **Customer Interaction Features:** The system should provide a way for customers to view the menu, select add-ons, and make informed decisions through features such as dish photos and detailed descriptions.

3 Theoretical Model of the System (System Design)

3.1 Defining System Requirements

Functional Requirements

1. **User-Friendly Admin Interface:**

   - **Menu Management Interface:** The system provides a form to add new dishes, including fields for dish type, name, price, description, and photo upload.
   - **Dish Type Management:** There is a section to add new dish types, update existing types, and delete dish types.
   - **Reorder Dish Types:** Admin can drag and drop dish types to reorder them.

2. **Responsive Design:** The interface is designed to be fully functional on both desktop and mobile devices. This ensures that users can manage the system efficiently, whether they are at a computer in the office or using a tablet or smartphone on the restaurant floor.

3. **Menu Management:**

- **Add New Dish:** Admin can add new dishes with a specified dish type, name, price, description, and photo, which will be displayed in the menu.
- **Edit and Delete Dishes:** Admin can update or delete existing dishes.
- **Filter Dishes by Type:** Admin can filter the displayed dishes by type and save the new order.
- **Add-ons Management:** Admin can add add-ons to dishes, including managing add-on pricing.

4. **QR Code Generation for Each Table:** The system will generate a unique QR code for each table in the restaurant. Customers can scan the QR code to access the menu, place orders, pay, and split the bill. This feature enhances convenience and streamlines the ordering process by reducing the need for physical menus and direct interaction with waitstaff.

5. **Waitstaff Interface:** A dedicated interface for waitstaff is essential, designed to be as simple and intuitive as possible. This interface will allow waitstaff to quickly create customer orders. The focus will be on minimizing the number of steps required to complete tasks, ensuring that the process is efficient even during peak hours. After analyzing relevant scientific literature "Design, Implementation, and Evaluation of a Menu Management System for Restaurants [6]", I decided to create the interface exclusively for use on phones and tablets, enabling waitstaff to move freely around the restaurant while managing orders.

6. **Customer Interface:** A user-friendly interface will be developed for customers, accessible through mobile devices via the QR code. This interface will allow customers to view the menu, select dishes and add-ons, place orders, pay their bill, and even split payments among the group. Designing this interface for mobile devices ensures that customers can interact with it conveniently from their phones or tablets while seated at their table.

Non-Functional Requirements

Usability: The interface is designed to be intuitive and easy to navigate for non-technical users. Both the waitstaff and customer interfaces will prioritize ease of use, ensuring that even users with minimal technological experience can efficiently interact with the system. The mobile and tablet optimization will ensure that the interface remains functional and user-friendly regardless of the device being used, thereby improving the overall dining experience.

This approach ensures that the system is accessible, efficient, and enhances the restaurant's operational workflow while providing a modern and convenient experience for both staff and customers.

3.2 Use case Diagram

3.2.1 Customer

The customer is an important user of the menu management system. They can access the system by either scanning a QR code on their table or by clicking a link to visit the website. Once on the site, customers can view the restaurant's menu without needing to log in. If logged in, they can also make a reservation for a table and write reviews about their dining experience. The QR code specifically helps customers access their table information directly. The system is easy to use on both smartphones and tablets, making it convenient for customers.

3.2.2 Waiter

The waiter uses a simple and easy-to-use interface designed for smartphones and tablets. They can create and manage orders quickly, which is especially useful during busy times. The waiter can make orders, add or remove items, make comments and manage addons for them.

3.2.3 Admin

The admin is responsible for managing the system. They can add, edit, or delete menu items and manage user roles. The admin also generates QR codes for tables, allowing customers to access the menu easily. In addition to these tasks, the admin can access a page where orders are displayed for the kitchen, ensuring that the cooking staff is aware of all incoming orders. The admin can also view and delete customer reviews, reservations, and user accounts, as well as create new waitstaff accounts. Furthermore, the admin has the ability to create add-ons and assign them to specific dishes.

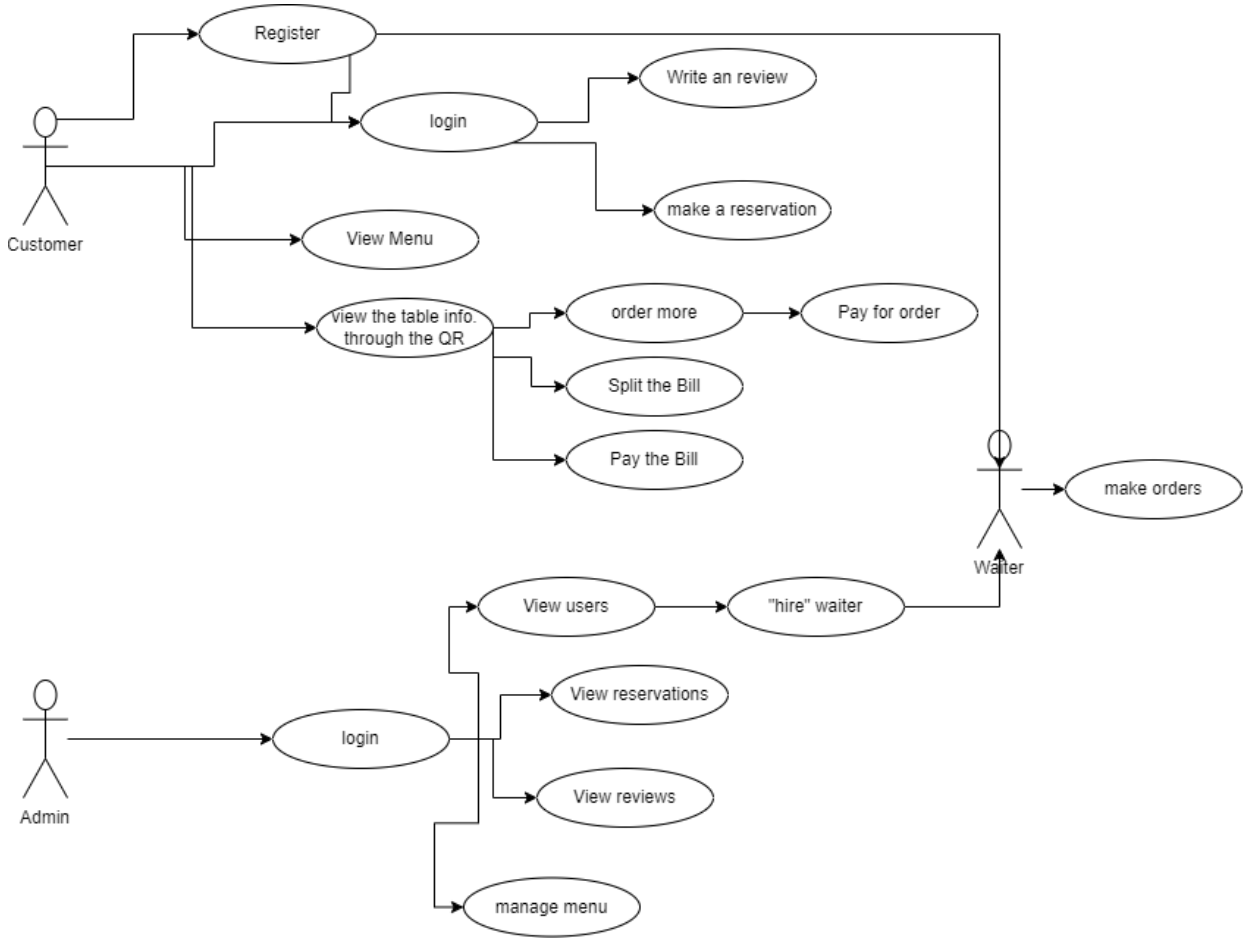

Figure 3. Use case diagram

4 Practical Implementation of the System

4.1 Technology Selection

For the implementation of the restaurant menu management system, I chose Laravel as the primary framework for several reasons. Laravel offers a comprehensive set of features, including a pre-built authentication system that simplifies the process of user login and registration. This feature is particularly valuable as it saves development time and ensures that the system is secure and reliable. Laravel's extensive documentation is another significant advantage, providing clear guidance and support, which is especially beneficial for developers who are implementing complex systems. Additionally, Laravel is widely recommended in academic and professional circles, including in the coursework I analyzed[1], which further reinforced my decision to use it for this project.

To handle payments within the system, I integrated **Stripe**, a powerful and flexible payment processing platform that allows secure and efficient transactions. Stripe's robust API made it easy to implement various payment methods, ensuring a seamless experience for the users.

For generating QR codes, which are used to access table-specific menus, I utilized the Laravel package l**simple-qrcode**l. This package provides an easy-to-use interface for creating QR codes, which are essential for enabling customers to quickly and conveniently access their table's menu via their smartphones.

Moreover, I utilized PHP for server-side scripting, handling backend logic, and JavaScript for client-side interactivity. For responsive design and styling, Bootstrap was employed, ensuring that the system functions well on various devices, including tablets and smartphones. I also integrated several libraries, such as SortableJS for drag-and-drop functionality, Font Awesome for icons, and DataTables for managing tabular data, to enhance the overall user experience and system efficiency. These technologies collectively contribute to creating a robust, scalable, and user-friendly restaurant management system.

For the implementation of the restaurant menu management system, the following technologies have been chosen:

1. **Languages**:
    - **PHP**: For server-side scripting and handling backend logic.
    - **JavaScript**: For client-side interactivity and DOM manipulation.
    - **CSS**: for styling specific components of the application
2. **Frameworks**:
    - **[5]Laravel**: PHP framework for building robust and scalable backend applications.
    - **[2]Bootstrap**: CSS framework for responsive design and styling.
3. **Libraries**:
    - **[7]SortableJS**: Provides drag-and-drop functionality for reordering elements.
    - **[4]Font Awesome**: Icon library for adding icons.
    - **[3]DataTables**: Library for managing tabular data, making it easier to search, sort, and paginate through large sets of menu items (planned).

4.2 System Architecture Description

The restaurant menu management system follows the MVC (Model-View-Controller) architectural pattern:

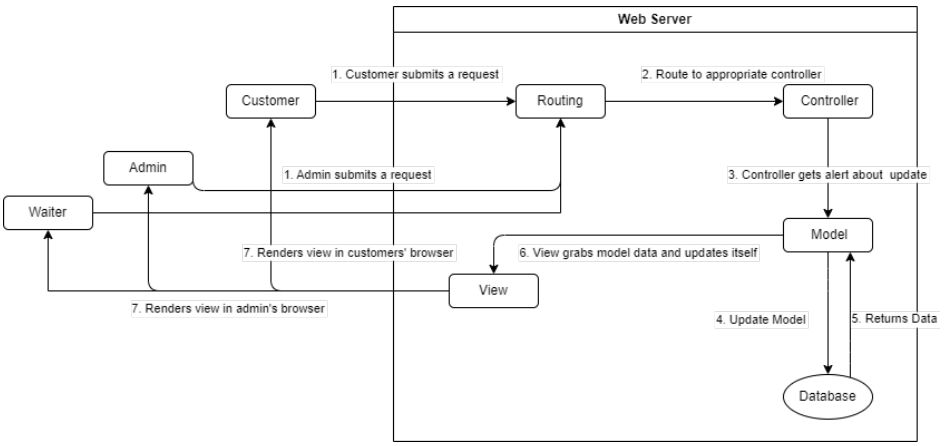

Figure 4. System Architecture

1. **Model**: Manages data and business logic.
    - Interacts with the database to retrieve, save, and update data.
    - Example models: Dish, DishType, Review, Reservation.
2. **View**: Responsible for presenting data to the user.
    - Uses Blade templating engine to render HTML.
    - Incorporates Bootstrap for responsive design.
3. **Controller**: Handles user requests and interacts with the Model.
    - Processes form submissions and updates the Model.
    - Returns the appropriate View with necessary data.
    - Example controllers: AdminController, MainController.

4.3 Database Implementation

The database for the restaurant menu management system was implemented using Laravel's powerful ORM (Eloquent) and migration features. Laravel simplifies the process of database creation and management, making it easier to define and organize tables, columns, and relationships. Each table in the system includes standard `created_at` and `updated_at` timestamp fields, which are automatically managed by Laravel to track when records are created and modified.

Table Structure and Division The database is logically divided into two main sections:

1. **Restaurant Ordering System:**
    - This part of the database handles all operations related to restaurant orders. It includes tables such as `dishes`, `dish_types`, `orders`, `add_ons`, and `tables`.

- These tables are closely linked together to ensure that orders, menu items, and add-ons can be managed efficiently. For example, the `orders` table stores details of each order and links to the `dishes` and `add_ons` tables through foreign keys. This allows the system to dynamically generate menus, manage add-ons, and ensure correct pricing and order handling.

2. **User and Customer Interactions:**

   - This part of the database is focused on user-related functionality. It includes tables for `users`, `reviews`, and `reservations`. These tables store user information, feedback, and reservation data, ensuring that customer interactions with the restaurant can be managed smoothly.
   - The `users` table, for example, was created using Laravel's built-in authentication features. This table stores user information such as names, emails, and encrypted passwords. The authentication system also automatically manages security features such as password hashing and user sessions.
   - The `csrf_token` is utilized throughout the application to protect against cross-site request forgery (CSRF) attacks. Laravel automatically generates and validates CSRF tokens for each user session, ensuring that forms submitted in the application are secure.

Relationships Between Tables This database is designed with clear relationships between tables, ensuring efficient data management and retrieval. Laravel's Eloquent ORM allows for the easy definition of these relationships, such as:

- **One-to-Many:** For example, a user can write multiple reviews, and the `reviews` table is linked to the `users` table through the `user_id` foreign key.
- **Many-to-Many:** The `orders` table can have multiple add-ons for each dish, and this relationship is managed through the `order_add_ons` pivot table.

Each of these relationships ensures that the data in the system remains consistent, allowing the application to operate efficiently and without data redundancy.

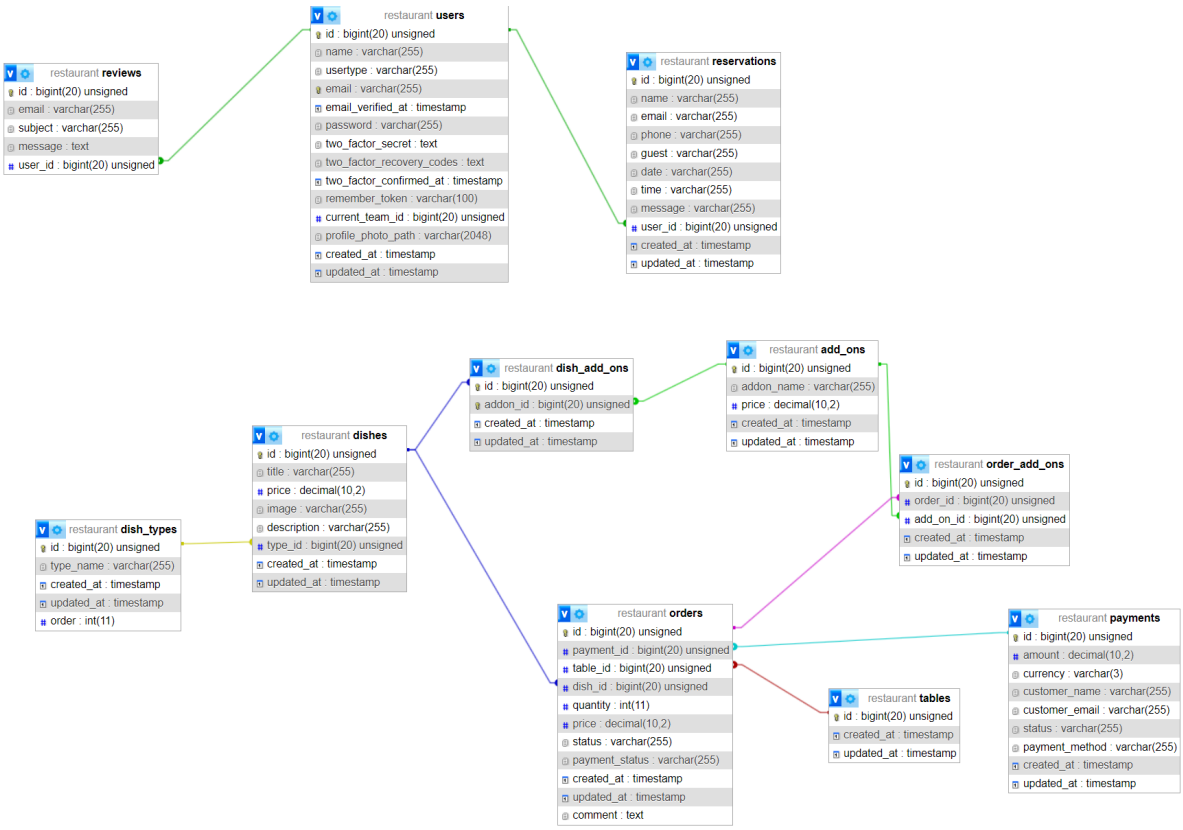

Figure 5. Relational schema

4.4 Problem Solving and Security Measures

Problem Solving

1. **Information Display on Mobile Devices**:
    - **Problem**: The website did not display information correctly on mobile devices.
    - **Solution**: Used Bootstrap's responsive grid system and media queries to ensure the layout adapts to different screen sizes.
2. **Sorting Food Types**:
    - **Problem**: Initially planned to change food types by swapping their IDs, which caused issues with connections to other tables.
    - **Solution**: Implemented a separate 'order' column to manage the sorting of food types without affecting their relationships with other tables and used the SortableJS library for drag-and-drop functionality.
3. **Order Status Handling:**
    - **Problem:** Initially, I used only a single status to track orders, such as "active," "completed" (after kitchen preparation), and "paid." However, since my system allows customers to pay in advance, this caused an issue where the order status would change to "paid" and disappear from the kitchen's view, even though the kitchen hadn't completed the order yet.

- **Solution:** To fix this, I introduced a separate "payment status" field to track payments, which allows the kitchen to still see orders marked as "active" until they are completed, regardless of whether the payment has been made or not.

4. **Additional Ordering by Customers:**
    - **Problem:** Initially, I wanted to allow customers to add new items to their order and pay for them later. However, I realized this could lead to a security issue where one person could add items to another table's order without paying for them.
    - **Solution:** To prevent this, I restricted the ability to add new items to orders to those already attached to the table and ensured that new items must be paid for immediately, thus preventing unauthorized additions to another customer's order.
5. **Real-time Order Updates for the Kitchen:**
    - **Problem:** The kitchen staff needed to view the most up-to-date orders in real-time, but the system required manual page refreshes to display new or updated orders. This led to delays in processing orders and increased the risk of missed updates.
    - **Solution:** To address this issue, I implemented an automated solution that regularly fetches the latest orders from the server without requiring the kitchen staff to manually refresh the page. This was achieved by using a background process that updates the order list every few seconds, ensuring that the kitchen always has the latest information on active orders. With this approach, kitchen staff can see new orders, any changes to current orders, or updates to order statuses in real time, improving efficiency and reducing the risk of errors caused by outdated information.
6. **Hosting on a Virtual Machine (Ubuntu)**:
    - **Problem**: Difficulty in deploying the server on a virtual machine running Ubuntu.
    - **Solution**: Followed a comprehensive guide from[8] [Code with Susan] to set up and deploy Laravel on Apache, ensuring a smooth and efficient deployment process.
7. **Database Schema Changes without Data Loss**:
    - **Problem**: Needed to update the database schema without losing existing data.
    - **Solution**: Used Laravel's migration feature to modify the database structure while preserving existing data.

Security Measures

1. **Data Encryption**: **Measure**: Encrypted sensitive data such as passwords using Laravel's hashing mechanisms.
2. **Input Validation and Sanitization**: **Measure**: Used Laravel's validation mechanisms to sanitize and validate user inputs.
3. **CSRF Protection**: **Measure**: Enabled Laravel's CSRF protection by including CSRF tokens in all form submissions.
4. **Authentication and Authorization**: **Measure**: Implemented Laravel's authentication system to secure user data and restricted access based on user roles.

5 Conclusions and Recommendations

5.1 Conclusions

1. **Key Observations**:
    - Effective use of Laravel's MVC architecture enhanced code organization and maintainability.
    - Bootstrap's responsive design ensured compatibility across various devices, improving user experience.
    - JavaScript, including SortableJS, was crucial for dynamic and interactive UI elements.
    - The integration of Stripe for payment processing provided a secure and efficient way to handle transactions, ensuring a smooth checkout process for users. Additionally, the use of the `simple-qrcode` package allowed for easy generation of QR codes, which are essential for the system's functionality in a restaurant setting.

5.2 Recommendations

Future Development:

- **Reservation Logic:** Develop comprehensive reservation management, including availability checks.
- **User Dashboard:** Fully update the user profile page to allow viewing and managing reservations, related orders, and customer reviews. This enhancement will involve linking the existing user interactions section of the database (which handles reviews and reservations) with the restaurant ordering system. This will create a cohesive user profile where customers can easily track their activities, including past and upcoming reservations, reviews, and order history.
- **File Management:** Implement features for deleting unused images and data files to optimize storage.
- **Tip Functionality:** Add the ability for users to leave tips for waitstaff when paying their bill. This feature will provide a more complete dining experience by allowing customers to directly reward good service through the system.

References

[1] Konstantinas Arefjevasl. *Maitinimo įstaigų valdymo sistema*, 2018.

[2] Bootstrap. *Bootstrap Documentation*. Retrieved from https://getbootstrap.com/docs.

[3] DataTables. *DataTables Documentation*. Retrieved from https://datatables.net/.

[4] Font Awesome. *Font Awesome Documentation*. Retrieved from https://fontawesome.com/docs.

[5] Laravel. *Laravel Documentation*. Retrieved from https://laravel.com/docs.

[6] Maxat Pernebayev, Tanvi Singh, and Karthick Sundararajan. *Design, Implementation, and Evaluation of a Menu Management System for Restaurants*. Retrieved from https://files.ifi.uzh.ch/CSG/staff/tsiaras/Extern/Theses/MP_Maxat_Pernebayev_Tanvi_Singh_Karthick_Sundararajan.pdf.

[7] SortableJS. *SortableJS Documentation*. Retrieved from https://sortablejs.github.io/Sortable/.

[8] Code with Susan. Deploy laravel on apache, n.d. Retrieved from https://codewithsusan.com/notes/deploy-laravel-on-apache.