Applied Information Technology and Computer Science Vol. 5 No. 1 (2024) 744-763

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

The Development of VGL Catering Booking System: VGL Cate using object-oriented approach

Vaani Letchumanan[1], Rosmamalmi Mat Nawi[1*]

[1]Fakulti SainsKomputerdanTeknologiMaklumat,
UniversitiTun Hussein Onn Malaysia, Parit Raja, BatuPahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2024.05.01.042
Received 24 June 2023; Accepted 18 may 2024; Available online 30 August 2024

Abstract: The VGL Catering Booking System addresses the need for a systematic booking management system for VGL Catering. This project aims to design and develop a comprehensive catering booking management system for VGL Catering, integrating a more systematic method like the E-Catering system. The system will be presented as a new branded online platform dedicated to handling bookings for VGL Catering. The objectives include implementing login, food ordering, date reservation, and data management modules, ensuring system functionality and usability. The waterfall model is employed as the methodology for developing the system. The main problem this system solves is the difficulty faced by customers in checking VGL Catering's availability due to the absence of a proper booking system. Additionally, management faces challenges in organizing and managing catering bookings effectively. The system will enable customers to reserve dates in advance by paying a deposit via online payment, even without placing a food order. The proposed system, VGL Cate, is designed based on an object-oriented approach, with the main feature being the booking system. By utilizing the VGL Cate system, customers can easily make decisions when booking food and catering services.

Keywords: Catering Booking System, Online Booking System, E-Catering, date reservation, online payment

1. Introduction

The booking system for VGL Catering will present as new management system in order to handle the bookings for the catering services. The main purpose of this system is to allow the customers to place their bookings through online. This system also will allow the management to manage the online transactions like invoices and receipts. All the data will used to generate the monthly report, which can allow the management to view the summary of catering bookings.

The main objective of this project is to develop a Catering Booking system: VGL Cate for VGL Catering using an object-oriented approach. The VGL Catering System, known as VGL Cate, aims to provide a systematic solution for both the organization and customers in managing catering bookings.

**Corresponding author: rosmamalmi @uthm.edu.my*

The main feature of the proposed system, VGL Cate is the ability for customers to book a specific date and select their desired food options for catering services. The system also includes a payment module to assist the organization in managing invoices and keeping track of income. Additionally, the booking system within VGL Cate helps the organization create and maintain a proper schedule for managing bookings. Customers can conveniently pay the required deposit through the system.

The primary users of this system will be the customers of VGL Catering and the administrative staff. The proposed system, VGL Cate will be a web-based system and available in English language. It comprises six modules, including registration, login, product, booking, payment and report module. Each module serves a specific purpose in facilitating the booking process, providing necessary information, managing payments, and generating reports for record-keeping.

The article organized into five sections. The first part is an introduction describing the context of the project. The second section describes the related work for the proposed system. The third section explains about the methodology that used to develop the proposed system including the analysis and design. The following section contains about the results and discussion of the system. Finally, the last section concludes the current work and highlights the future works.

2. Related Work

2.1 Online Booking System

An online booking system is a software solution and reservation system that simplifies the process for customers to reserve and pay for their activities [1]. Implementing such a system brings ease and convenience to the booking process, ultimately resulting in increased bookings for the organization. Furthermore, a booking system goes beyond merely accepting online bookings and payments; it can offer additional functionality to customers [2]. With a potential online booking system, customers will have the ability to self-book and make payments directly through the website [3]. Additionally, the system will securely store data and information, enabling efficient management of bookings from the comfort of one's home and ensuring the continuity of business operations over an extended period [3].

2.2 Web-based system

Web-based systems have grown in popularity in recent years because of their numerous benefits. a web-based system offers numerous advantages in terms of accessibility, scalability, and ease of use [4]. A web-based system allows for seamless integration with various platforms and devices, enabling users to access the system from anywhere at any time [5]. the cost-effectiveness of web-based systems, as they eliminate the need for extensive hardware infrastructure and can be easily updated and maintained [6]. Web-based systems enhance collaboration and information sharing among users, leading to improved efficiency and productivity in organizational settings. A web-based system is notable for giving users access to real-time data, which offers online reports, and digital dashboards can replace spreadsheet reports [7].

2.3 Comparison of Equivalent System

Three existing system related to current developing system were analyzed and researched to know what to develop in the proposed system, and what features should be included to develop a better, user-friendly catering booking system. Three existing systems reviewed and compared are Lynn’s Catering system, Surya Indian Kitchen N Caterers and Indian Accent.

The proposed system, VGL Cate offers competitive advantages over other systems by being more accessible to customers who prefer an online booking system. It includes a registration and login module, allowing for easy tracking of customer information and booking history. Unlike other systems,

VGL Cate features a date pre-reservation module which allows the customers to book the date by paying a specific amount as a deposit through the system. The system also provides online payment and invoice and receipt generation, enhancing convenience for customers and streamlining the payment process.

The delivery service, present in all systems except Lynn's Catering, enables customers to enter their event location in order to receive the ordered food items. Overall, VGL Cate comprehensive and efficient system meets customer needs and simplifies administration processes. Table 1 shows the comparison table between the existing systems and the proposed system.

Table 1: The comparison table between the existing systems and the proposed system

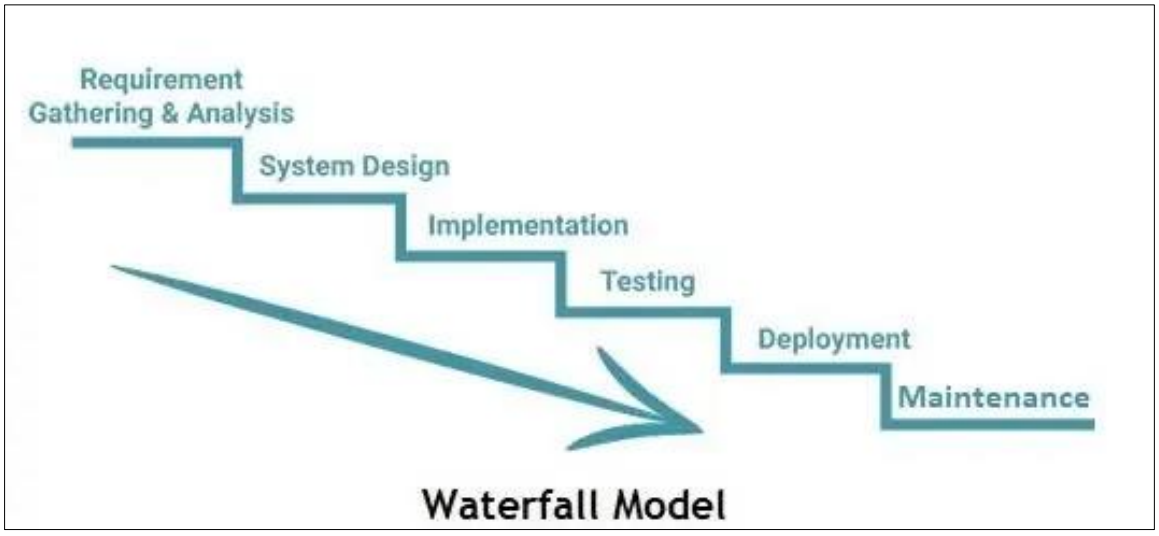
Features/System	Lynn's Catering	Surya Indian Kitchen N Caterers	Indian Accent Catering system	VGL Cate: Catering Booking System
System Type	Web-based	Web-based	Web-based	Web-based
Login Module	X	X	X	√
Registration Module	X	X	X	√
Date Pre-reservation	X	X	X	√
Booking Module	√	√	√	√
Order Food	√	√	√	√
Customizable menu	X	X	√	√
Invoice and receipt	X	X	X	√
Online Payment	X	X	X	√
Delivery service	X	√	√	√
Generate Report	X	√	X	√

3. Methodology

3.1 Waterfall Model

The waterfall model has specified that only should go on to the next step once all prior stages have been thoroughly tested, reviewed, and confirmed. the waterfall model was originally presented as a flawed approach [8]. Beginning with feasibility and moving through numerous tasks until actual deployment, activities conducted in a sequential order. Requirement gathering and analysis flow into system design, which feeds into implementation, followed by testing, deployment and finally maintenance. The model's rigid and inflexible nature, with limited opportunities for iteration and changes, has been a subject of criticism in software engineering literature. Despite its limitations, the waterfall model is still used in certain projects with well-defined and stable. Because the testing process occurs towards the end of the model, it has been difficult to have feeds passed back up the waterfall. A waterfall model shown in Figure 1.

Figure 1: Waterfall Model



3.2 System Development Workflow

In the requirement gathering and analysis stage, the admin was interviewed to describe the module, functionalities, and other requirements that are required in the system, as well as the challenges that produced by the current technique. This has done to guarantee that a better-qualified system will be developed. The requirements for the project fully analyzed. The purpose of the analysis is to determine which system components, such as hardware, software, networks, and human resources, are currently in use. The functional requirements, non-functional requirements specified in this phase.

In design phase, the user interfaces for the VGL Catering system designed appropriately with the user requirements such as the designs of the booking module and its procedure to book an order with the system architecture. The Gantt Chart, UML diagram, and a Class Diagram designed in this phase. In implementation phase, the development of VGL Cate system begin. The system's code written in the Java programming language. The database will be set up using MySQL and connected to the system's user interface. The prototype for the system created based on the information acquired throughout the planning and system design phases.

The next phase in this methodology is testing and deployment phase. In this stage, the prototype and report will be generated. The functions and features of VGL Cate system will be checked and tested by the client. Commentary will be given by the client to improvise the project from any angle and the bugs will be fixed until the client is satisfied with the entire system. Then the system will be finalized using website and test cases. Table 2 shows the software development activities and their task.

Table 2: Software development activities and their task

Phase	Task	Output
Requirement Gathering & Analysis	• Study of existing system • Comparison among three existing system and the proposed system, • Do interview with VGL Catering owner and do survey related to catering booking system	• Functional & non-functional requirements • Software and hardware requirements • Literature review

	• Analyse the result of interview and survey • Define hardware and software requirement to develop VGL Catering system	
System Design	• Design process system, database system, and interfaces of the proposed system • Create wireframe for user interface • Illustrate use case diagram, sequence diagram and overall system architecture	• Use Case Diagram and Class Diagram • User interface • Database design
Implementation	• Develop the prototype based on requirement analysis and prototype design	• Prototype of system
Testing	• Testing the system before handover to the organization	• Prototype of system
Deployment	• The proposed system was deployed	• Workable system
Maintenance	• Deploy the system • Fix the errors or bugs	• Workable system • Full report for the project

3.3 Requirement Analysis

Requirement analysis is the process of determine requirements that developed system needs to fulfil, or user expectation outcome from the proposed system. System requirements include functional and non-functional requirements, user requirements and system requirements. A functionality, operations, or activities that describe what task a system is going to execute can all considered as functional requirements and non-functional requirements allow users to make limits or prohibitions on the system's architecture throughout many iterative queues [9]. Table 3 shows the functional requirements and Table 4 shows the non-functional requirements of VGL Cate.

Table 3: Functional Requirement of VGL Cate

No.	Modules	Functionalities
1	Login Module	• The system should allow the users to login into the system using email and password. • The system should validate the information entered by users and alert for any invalid input • The system should redirect the user to respective pages, either are booking page or booking page once successful login.
2	Registration Module	• The system should allow the new user to register before login. • The system should validate the information entered by users and alert for any invalid input. • The system should save registration request will be stored for further approve.

3	Booking Module	• The system should allow the customer to place the booking.
4	Product Module	• The system should allow the admin to add, remove and update food items to the list.
5	Payment Module	• The system should show the quotation to the customer before they confirm with the booking. • The system should allow the customer to pay in online using payment gateway.
6	Report Module	• The system should let admin to generate a monthly report

Table 4: Non-Functional Requirement of VGL Cate

No	Requirements	Description
1.	Performance	• The system should have smooth and fast functions
2.	Operational	• The system should be updated and maintained well. • The system should be user friendly.
3.	Security	• There will no unauthorized users logging into the system.
4.	Reliability	• The system should work well even if the site is used extensively
5.	Usability	• The system will not have a difficulty while being used to operate certain functions

3.4 System Analysis

The system analysis aims to obtain overall and throughout meaning and main idea of the system environment. To perform system analysis, Object-Oriented Programming approach is used. The results of the system analysis, a Unified Modelling Language (UML) diagram defined in this sub-section. Major elements of the UML Use Case Diagram of VGL Catering system shown in the Figure 2 below. Figure 2, states that the system has 2 users, admin and customer. Only the customer has to register an account in the system. The both users, admin and customer can log in and log out of the system. The customer can add a booking, view menu to add the food items and select the preferable date in order to proceed the booking. The admin can add and update the food product details, and update the available dates for booking. The invoice and order details can be viewed by both users, customers and admin.

Figure 2: Use Case Diagram of VGL Catering System

3.5 Overall System Architecture

This system architecture diagram is used to depict how each module in a system interacts with one another. Figure 3 shows the system architecture diagram of VGL Cate.

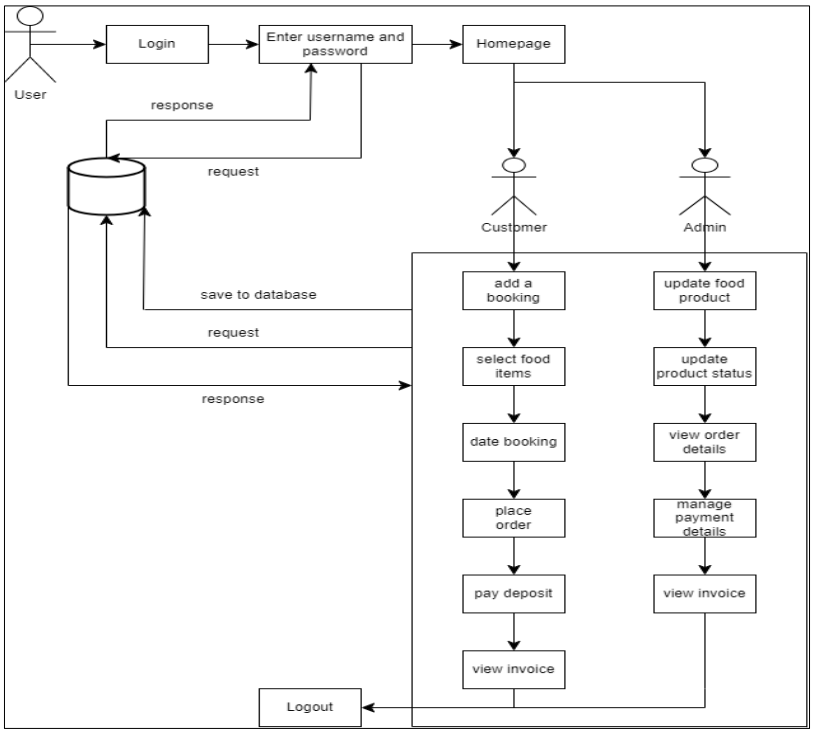


Figure 3: Overall System Architecture of VGL Cate

3.6 Class Diagram

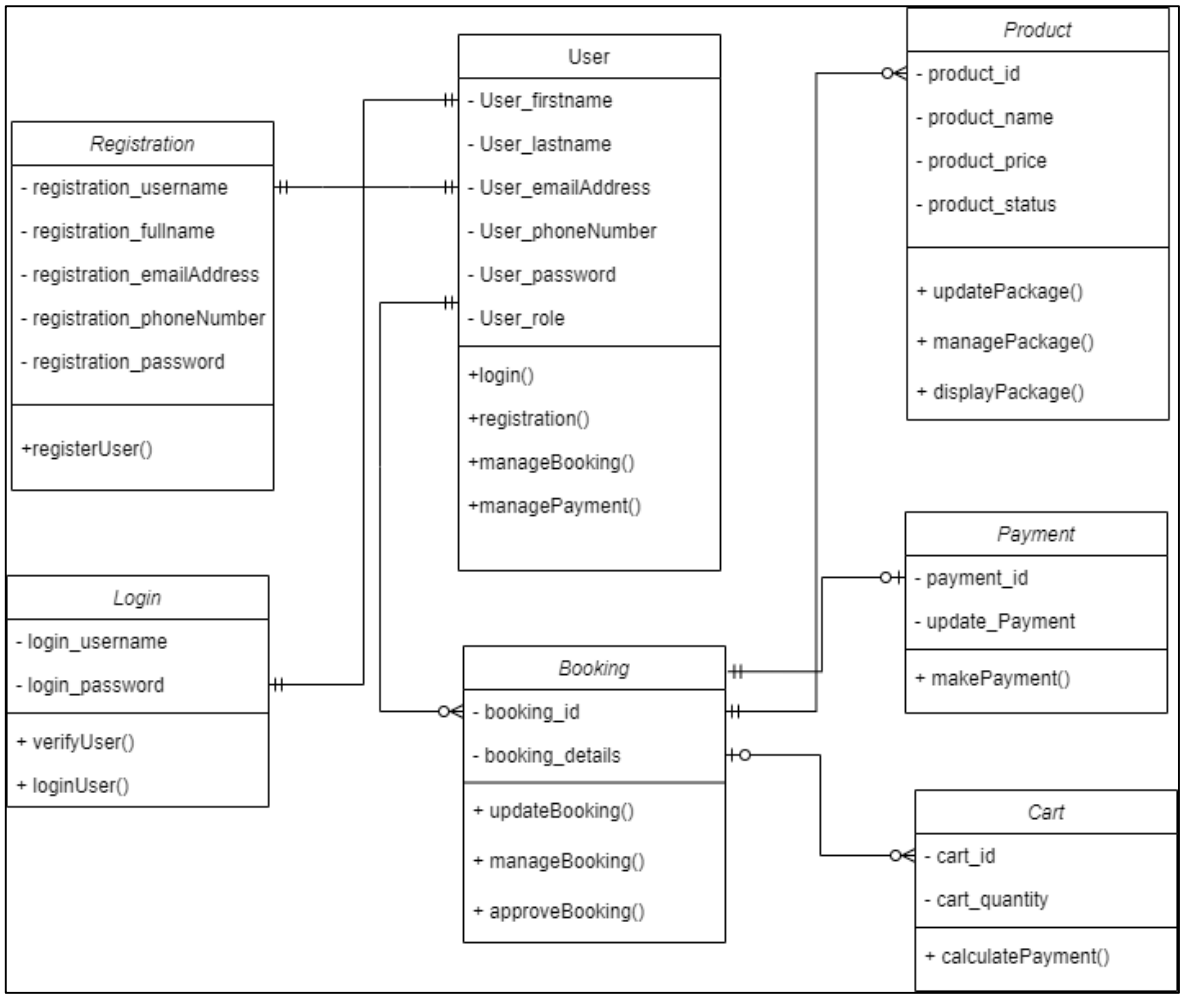


Figure 4: Class Diagram of VGL Catering System

3.7 Sequence Diagram

Sequence diagrams are used to describe the interactions and behaviors that take place between actors and objects. Only when necessary or when another object want to interact with them will actors or objects be brought into the scene. Figure 4 and Figure 5 show the sequence diagrams for each user, customer and admin.

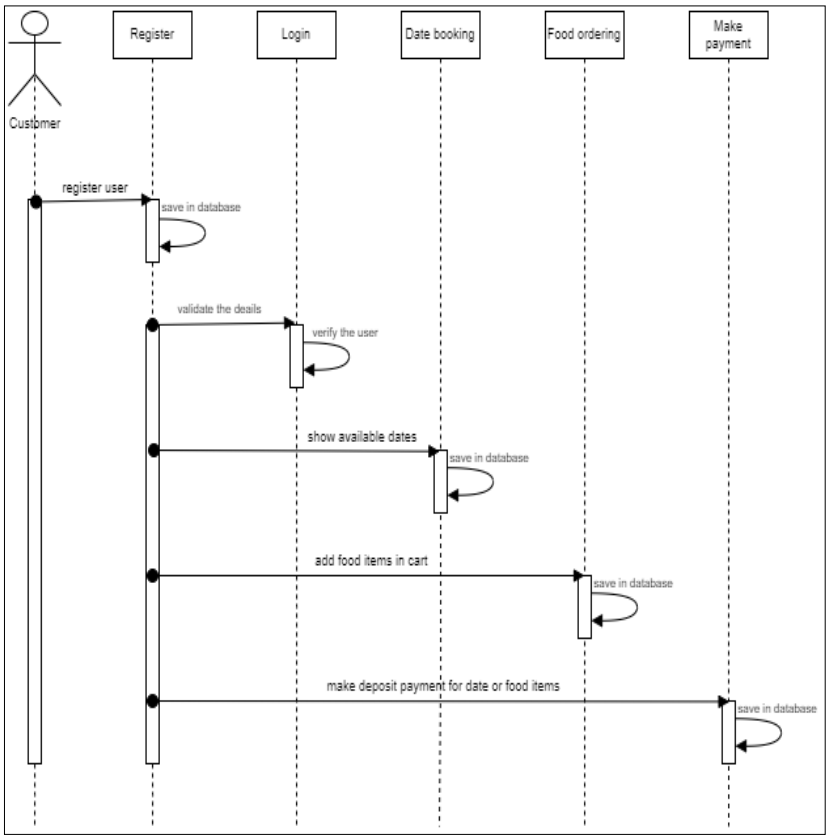


Figure 5: Sequence diagram for customer

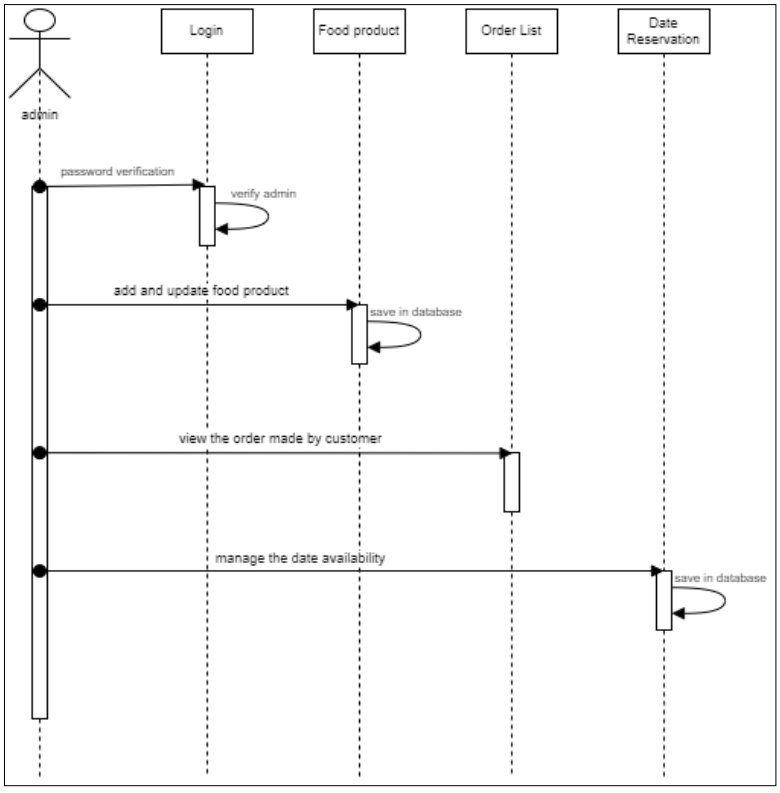


Figure 6: Sequence diagram for admin

3.7 Implementation

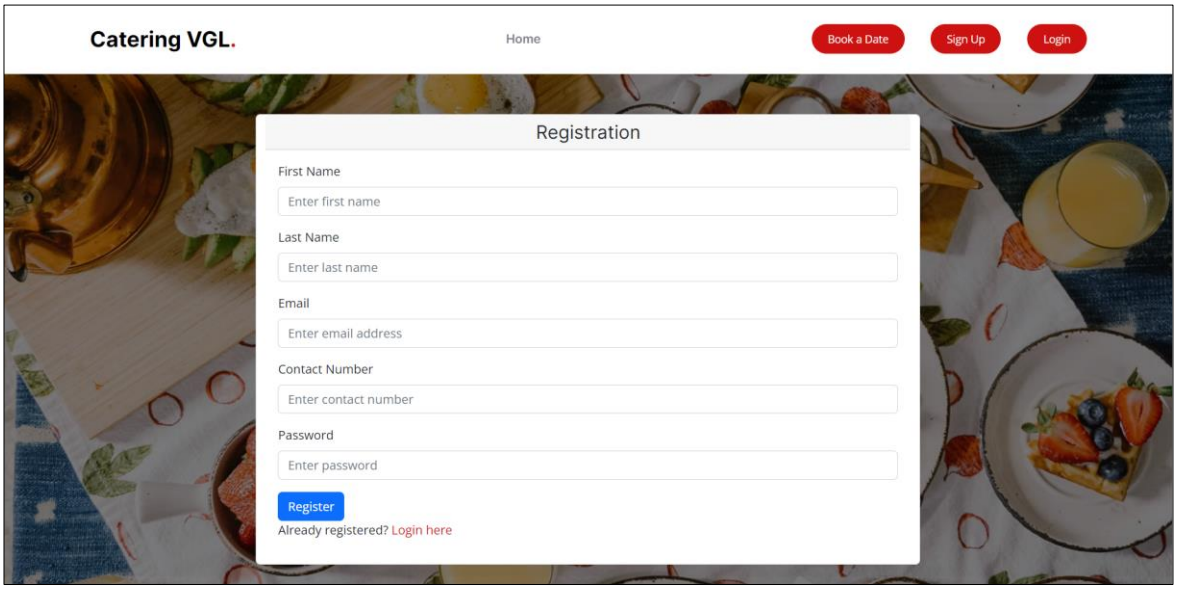


Figure 7: Register Page for both users

```
@PostMapping("/doSave")
public String registration(@Valid @ModelAttribute("user") UserDto user, BindingResult result, Model model) {
    User existing = userService.findByEmail(user.getEmail());
    if (existing != null) {
        result.rejectValue("email", null, "There is already an account registered with that email");
    }
    if (result.hasErrors()) {
        model.addAttribute("user", user);
        return "user/register";
    }
    userService.saveUser(user);
    return "redirect:/register?success";
}
```

Figure 8: Code segment for Registration Page both users

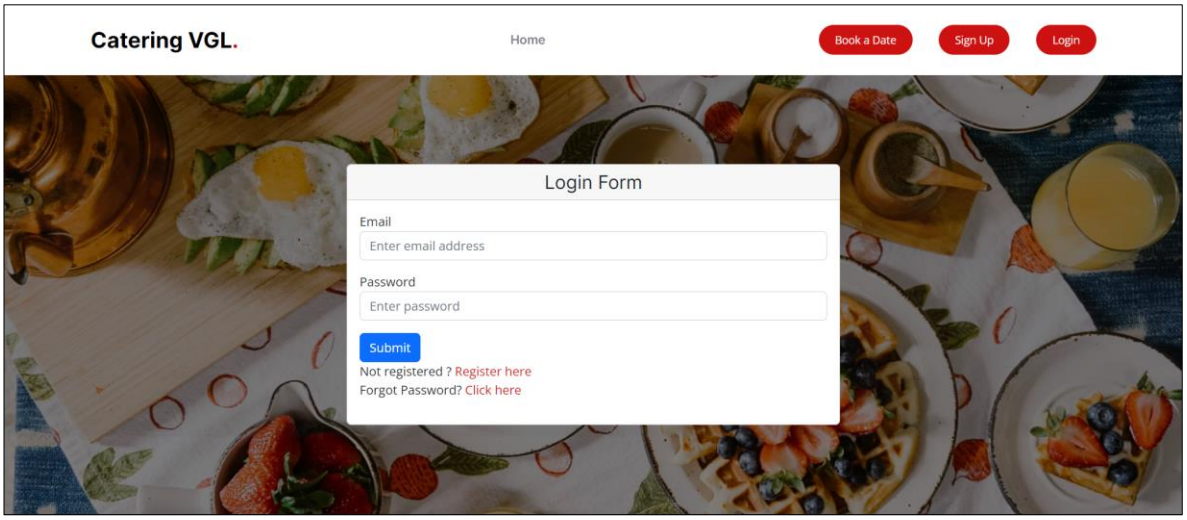


Figure 9: Login Page for both users

```
@GetMapping("/login")
public String loginForm() {
    return "user/login";
}
```

Figure 10: Code segment for Login Page both users

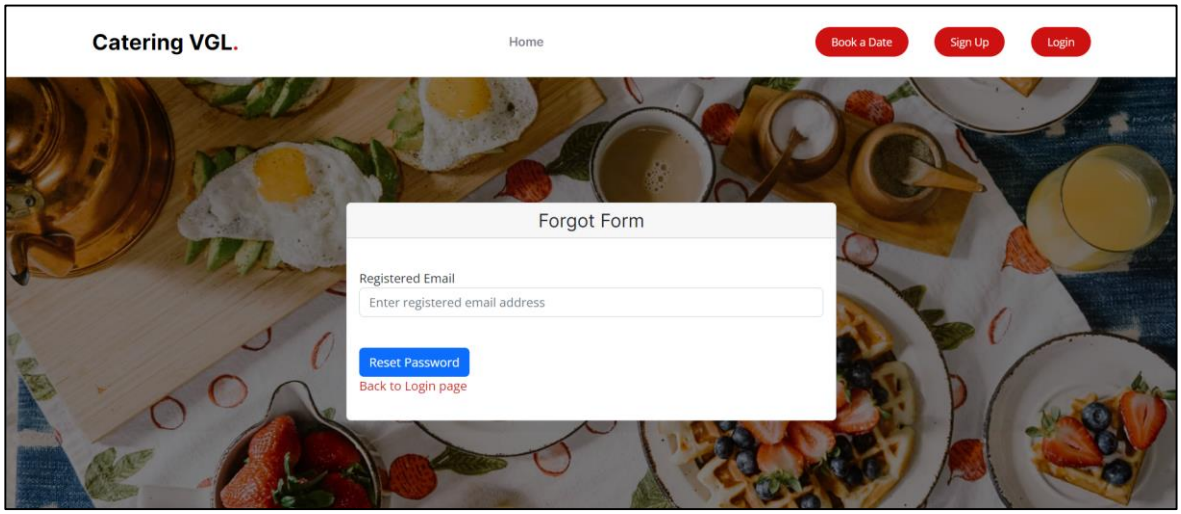


Figure 11: Forgot Password Page for both users

```
@PostMapping("/doForgotPassword")
public String forgotpassword(@Valid @ModelAttribute("user") UserForgetPasswordDto userDto, BindingResult result,
        Model model) {

    try {

        if (userDto.getEmail().isBlank()) {
            // result.rejectValue("email", null, "Please provide valid input");
        } else {
            User user = userRepo.findByEmail(userDto.getEmail());
            if (user != null && !Strings.isNullOrEmpty(user.getEmail())) {
                emailUtils.sendForgotMail(user.getEmail(), "Credential by VGL System", user.getPassword());
                return "redirect:/auth/forgotPassword?success";
            } else {
                result.rejectValue("email", null, "User not found");
            }
        }

        if (result.hasErrors())
            model.addAttribute("user", userDto);

    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return "/user/forgotpassword";
}
```

Figure 12: Code segment Forgot Password Page for both users

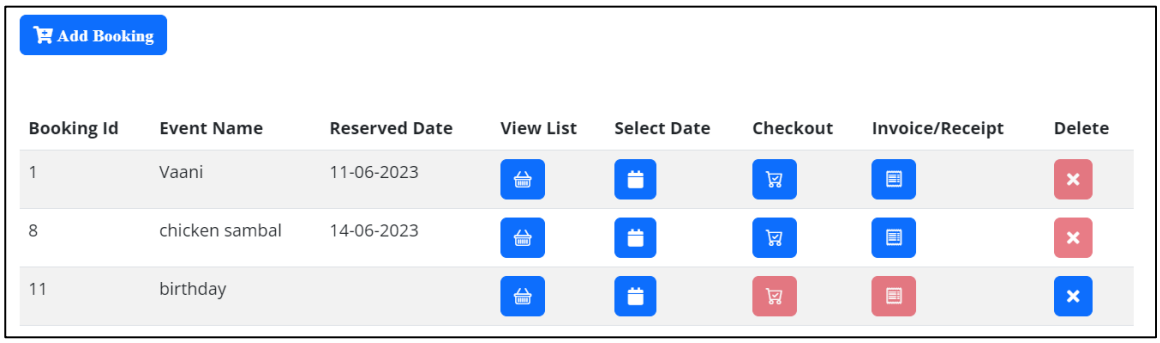


Figure 13: Booking Page for customer side

```java
@GetMapping("/listing")
public ModelAndView bookingListing(Principal principal) {
    // Get all bookings based on logged in user
    User user = userRepo.findByEmail(principal.getName());
    ModelAndView mav = new ModelAndView("booking/booking");
    List<Booking> bookings = bookingRepo.getAllBooking(user);

    List<BookingDto> bookingsDto = new ArrayList<BookingDto>();
    for (Booking booking : bookings) {

        List<Cart> carts = cartRepo.getCartByBookingId(booking.getId());

        BookingDto bookingDto = new BookingDto();
        bookingDto.setCartSize(carts.size());
        bookingDto.setId(booking.getId());
        bookingDto.setName(booking.getName());
        bookingDto.setRsvpDate(booking.getRsvpDate());

        bookingsDto.add(bookingDto);
    }

    mav.addObject("bookings", bookingsDto);
    return mav;
}
```

Figure 14: Code segment for booking page for customer side

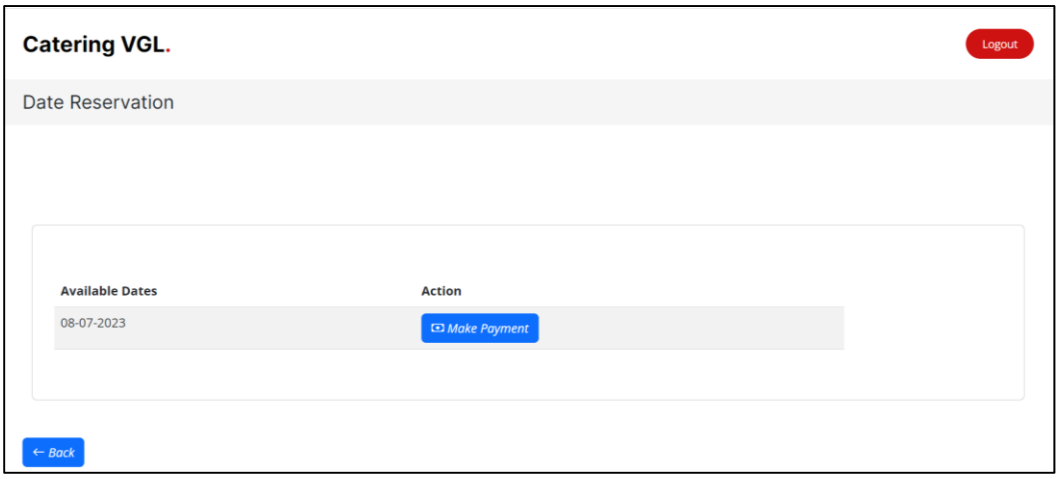


Figure 15: Date Pre-Reservation page for customer side

```java
@RequestMapping("/paymentSuccess/{bookingId}/{rsvpDateId}")
public String paymentSuccess(@PathVariable("bookingId") String bookingId, @PathVariable("rsvpDateId") String rsvpDateId,
        Model model) {

    long bId = Long.parseLong(bookingId);
    long dId = Long.parseLong(rsvpDateId);

    RsvpDate rsvpDate = rsvpDateRepo.getReferenceById(dId);
    Booking booking = bookingRepo.getReferenceById(bId);


    booking.setRsvpDate(rsvpDate);
    bookingRepo.save(booking);

    rsvpDate.setAvailable(Boolean.FALSE);
    rsvpDateRepo.save(rsvpDate);

    return "redirect:/booking/listing";
}
```

Figure 16: Code segment for date pre-reservation page

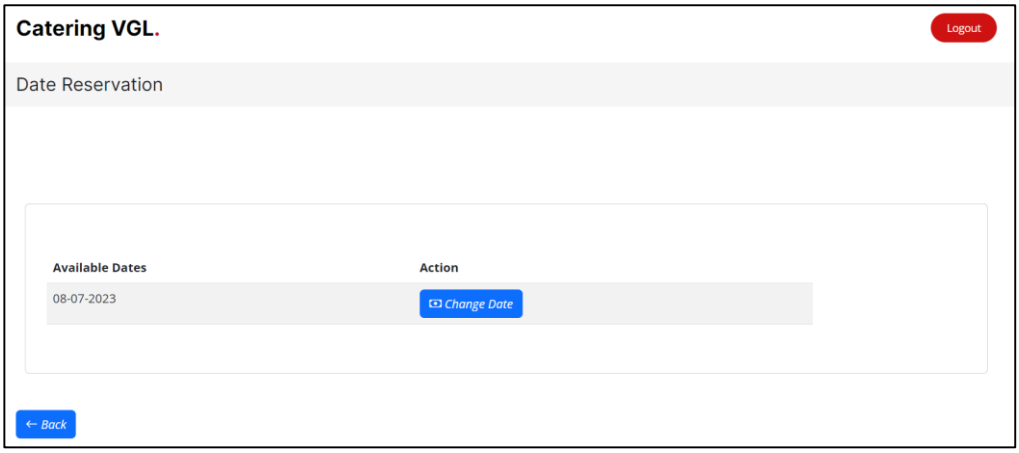


Figure 17: Change date page for customer side

```
@RequestMapping("/changeDate/{bookingId}/{rsvpDateId}")
public String showDateChangingPage(@PathVariable("bookingId") String bookingId, @PathVariable("rsvpDateId") String rsvpDateId,
        Model model) {

    long bId = Long.parseLong(bookingId);
    long dId = Long.parseLong(rsvpDateId);

    RsvpDate newRsvpDate = rsvpDateRepo.getReferenceById(dId);
    Booking booking = bookingRepo.getReferenceById(bId);

    RsvpDate oldRsvpDate = booking.getRsvpDate();
    oldRsvpDate.setAvailable(Boolean.TRUE);
    rsvpDateRepo.save(oldRsvpDate);

    booking.setRsvpDate(newRsvpDate);
    bookingRepo.save(booking);

    newRsvpDate.setAvailable(Boolean.FALSE);
    rsvpDateRepo.save(newRsvpDate);

    return ("redirect:/booking/listing");
}
```

Figure 18: Code segment for change date page

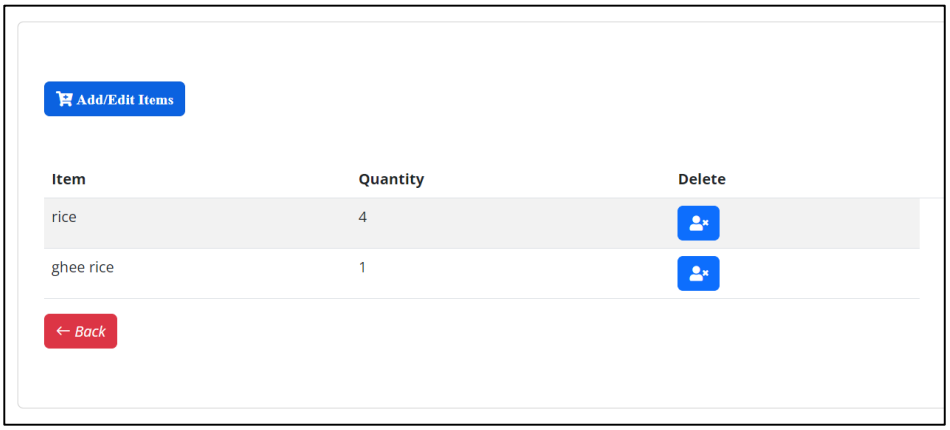


Figure 19: Cart page for customer side

```
@PostMapping("/addToCart")
public String addToCart(@ModelAttribute("menuDto") MenuDto menuDto) {

    Booking booking = bookingRepo.getReferenceById(menuDto.getBookingId());
    List<Cart> carts = cartRepo.getCartByBookingId(menuDto.getBookingId());
    List<CartProductMenuDto> cartProductMenuDtos = menuDto.getCartProductMenuDtos();
    for (CartProductMenuDto cartProductMenuDto : cartProductMenuDtos) {

        int quantity = cartProductMenuDto.getQuantity();
        if (carts.size() != 0) {

            if (quantity != 0) {
                Cart tempCart = null;
                for (Cart cart : carts) {
                    if (cart.getProduct().getId() == cartProductMenuDto.getProductId()) {
                        tempCart = cartRepo.getReferenceById(cart.getId());
                        tempCart.setQuantity(quantity);
                        cartRepo.save(tempCart);
                    }
                }
                if (tempCart == null) {
                    Product product = productRepo.getReferenceById(cartProductMenuDto.getProductId());
                    Cart cart = new Cart();
                    cart.setProduct(product);
                    cart.setQuantity(quantity);
                    cart.setBooking(booking);
                    cartRepo.save(cart);
                }
            } else {
                for (Cart cart : carts) {
                    if (cart.getProduct().getId() == cartProductMenuDto.getProductId()) {
                        cartRepo.delete(cart);
                        break;
                    }
                }
            }
        } else {
            Product product = productRepo.getReferenceById(cartProductMenuDto.getProductId());
            if (quantity != 0) {
                Cart cart = new Cart();
                cart.setProduct(product);
                cart.setQuantity(quantity);
                cart.setBooking(booking);
                cartRepo.save(cart);
            }
        }
    }
    return "redirect:/showCart/" + menuDto.getBookingId();
}
```

Figure 20: Code segment for cart page for customer side

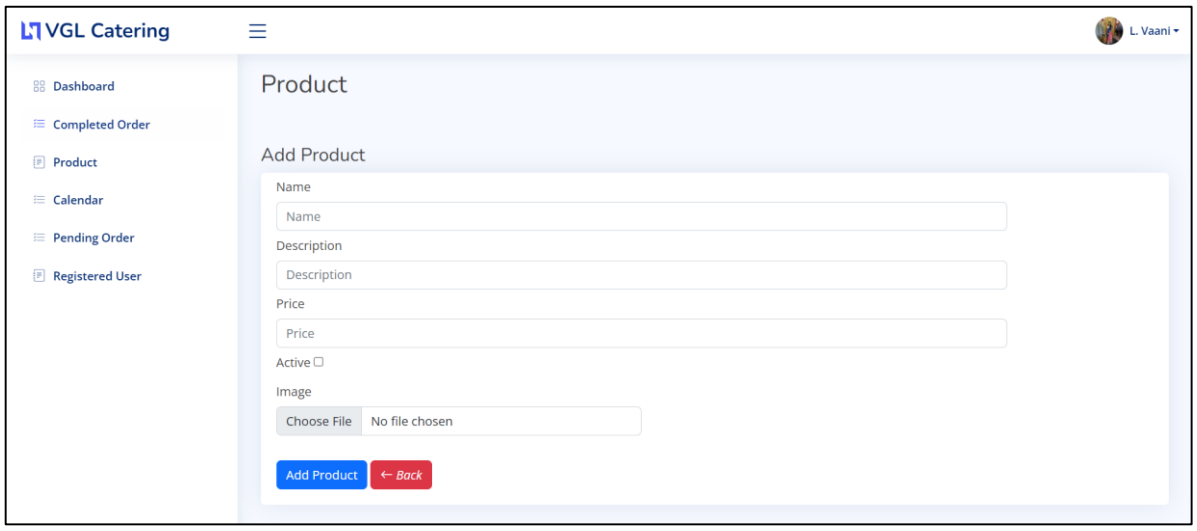


Figure 21: Add product page for admin side

```
@PostMapping("/doAdd")
public String doAddProduct(@Valid ProductDto productDto, BindingResult result, Model model,
        @RequestParam("image") MultipartFile file, HttpServletRequest request) throws IOException {

    if (result.hasErrors()) {
        List<?> err = result.getAllErrors();
        System.err.println("Err is" + err);
        return "redirect:addProduct?nameMandatory";
    }

    String uploadDirectory = request.getServletContext().getRealPath(uploadFolder);
    System.err.println("uploadDirectory:: " + uploadDirectory);
    String fileName = file.getOriginalFilename();
    String filePath = Paths.get(uploadDirectory, fileName).toString();
    System.err.println("FileName: " + file.getOriginalFilename());

    Product product = new Product();
    product.setName(productDto.getName());
    product.setDescription(productDto.getDescription());
    product.setPrice(productDto.getPrice());
    product.setActive(productDto.getActive());

    if (fileName == null || fileName.contains("..")) {
        model.addAttribute("invalid", "Sorry! Filename contains invalid path sequence \" + fileName");
    }

    try {
        File dir = new File(uploadDirectory);
        if (!dir.exists()) {
            System.err.println("Folder Created");
            dir.mkdirs();
        }
        // Save the file locally
        BufferedOutputStream stream = new BufferedOutputStream(new FileOutputStream(new File(filePath)));
        stream.write(file.getBytes());
        stream.close();

        byte[] imageData = file.getBytes();
        product.setImage(imageData);

    } catch (Exception e) {
        System.err.println("in catch");
    }

    productRepository.save(product);

    return "redirect:/product/listing";
}
```

Figure 22: Code segment for add product page

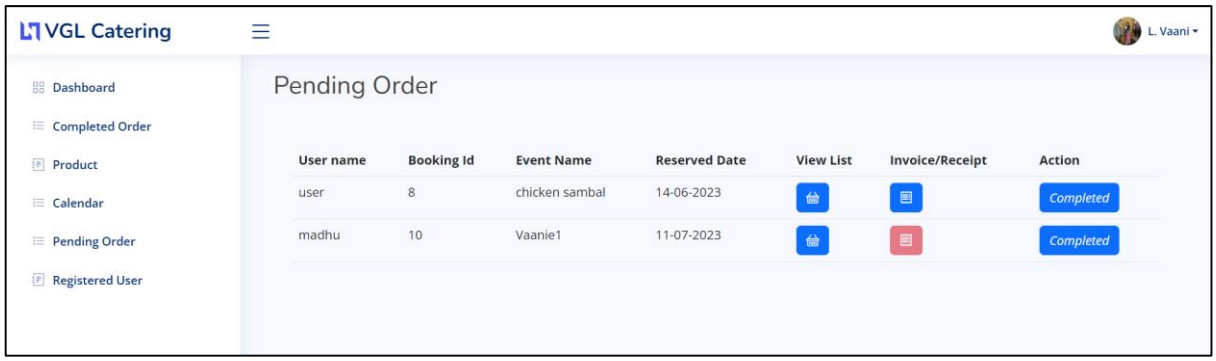


Figure 23: Pending Order for admin side

```java
@GetMapping("/pendingOrder")
public ModelAndView adminListing() {

    ModelAndView mav = new ModelAndView("pendingOrder/list");
    List<Booking> bookings = bookingRepo.findAll();

    List<BookingDto> bookingsDto = new ArrayList<BookingDto>();
    for (Booking booking : bookings) {

        if (booking.getRsvpDate() != null) {

            if (booking.getCompleted() == null || booking.getCompleted() == false) {
                List<Cart> carts = cartRepo.getCartByBookingId(booking.getId());

                BookingDto bookingDto = new BookingDto();
                bookingDto.setCartSize(carts.size());
                bookingDto.setId(booking.getId());
                bookingDto.setName(booking.getName());
                bookingDto.setRsvpDate(booking.getRsvpDate());
                bookingDto.setUser(booking.getUser());

                bookingsDto.add(bookingDto);
            }
        }
    }
    mav.addObject("bookings", bookingsDto);
    return mav;
}
```

Figure 24: Code Segment for Pending Order for admin side

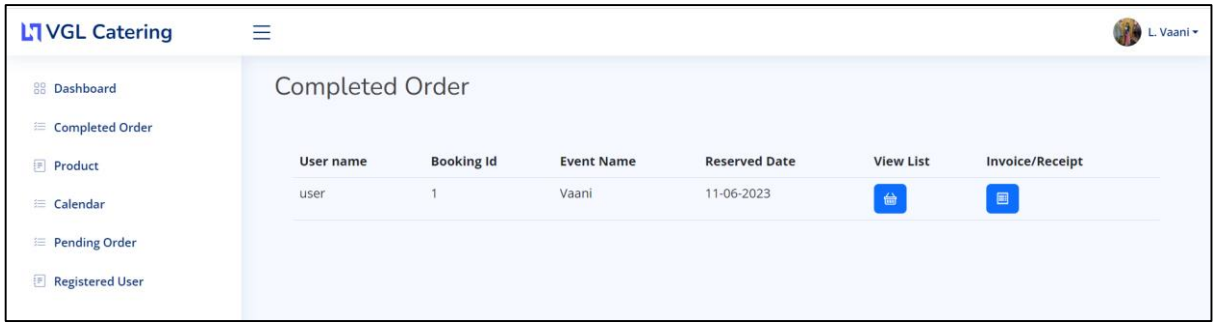


Figure 25: Completed Order page for admin side

```java
@GetMapping("/completedOrder")
public ModelAndView completedOrder() {

    ModelAndView mav = new ModelAndView("completedOrder/list");
    List<Booking> bookings = bookingRepo.findAll();

    List<BookingDto> bookingsDto = new ArrayList<BookingDto>();
    for (Booking booking : bookings) {

            if (booking.getCompleted() != null &&  booking.getCompleted()) {
                List<Cart> carts = cartRepo.getCartByBookingId(booking.getId());

                BookingDto bookingDto = new BookingDto();
                bookingDto.setCartSize(carts.size());
                bookingDto.setId(booking.getId());
                bookingDto.setName(booking.getName());
                bookingDto.setRsvpDate(booking.getRsvpDate());
                bookingDto.setUser(booking.getUser());

                bookingsDto.add(bookingDto);
            }
    }
    mav.addObject("bookings", bookingsDto);
    return mav;
}
```

Figure 26: Code segment for Completed Order page for admin side

4. Results and Discussion

4.1 Testing

Testing is of utmost importance in the development of a web-based system. It ensures the stability and reliability of an app by addressing and fixing bugs and errors. Table 6 shows the test case for each module.

Table 5: Test Case

Test Case Id	Description	Expected Result	Status
	Register Module (TC_100)		
TC_100_01	Customers click the Signup button after entering all the information required for registration.	The system should prompt a registered successful message on the screen	PASS
TC_100_02	Customer register using an email address that already has been created before.	The system should prompt an invalid email message on the screen.	PASS
TC_100_03	Customers press the Sign Up button without filling in the required information for registration.	The system should ask the user to fill in all the details required.	PASS
	Login Module (TC_200)		
TC_200_01	Customers log in to the system without registering.	The system prompts an invalid email message on the screen.	PASS
TC_200_02	Customer login into the system after successfully registering an account.	The system should allow the customer to log into the system.	PASS
TC_200_03	Customer login into the system without entering a password or email.	The system should prompt an error message asking the user to enter their email and password on the screen.	PASS
TC_200_04	Customer login into the system by entering the wrong email or password.	The system prompts an error message asking the user to enter the email and password again on the screen.	PASS
TC_200_05	Admin login into the website by entering username and password.	The system shows a login successful message on the screen and redirects the admin to the homepage.	PASS
TC_200_06	Admin clicks login button by entering wrong username or password.	The system prompts an invalid email or password error message at the screen.	PASS
	Product Module (TC_300)		
TC_300_01	Customers click any services food products.	The system add the food items into cart list.	PASS

TC_300_02	Admin clicks add a product button to add a new product.	The system will ask the user to enter the necessary details to add a product	PASS
TC_300_03	Admin clicks the save button after entering all the details required.	The system will save the details of the product	PASS
TC_300_04	Admin deselect active status button on any products.	The system will hide the food product from the menu	PASS
Booking Module (TC_400)			
TC_400_01	Customers click confirm booking button after selecting their preferred date.	The application will redirect the customer to the payment page.	PASS
TC_400_02	Admin clicks the “Set Unavailable” button.	The system allow admin to block an unavailable date.	PASS
Payment Module (TC_500)			
TC_500_01	The user clicks the payment button after entering a specific amount of money for the deposit.	The system will redirect to the payment page where the user has to enter credit card details to proceed with the payment.	PASS
TC_500_02	The User clicks validate button after entering all the data needed for credit card payment.	The system will display the invoice with all the booking details.	PASS
TC_500_04	Admin clicks the “Completed” button in the system for fully paid customers.	The system will move the particular order to completed order list	PASS

4.2 User Acceptance Test

The user acceptance test for the VGL Catering application demonstrated positive feedback from 20 student respondents.

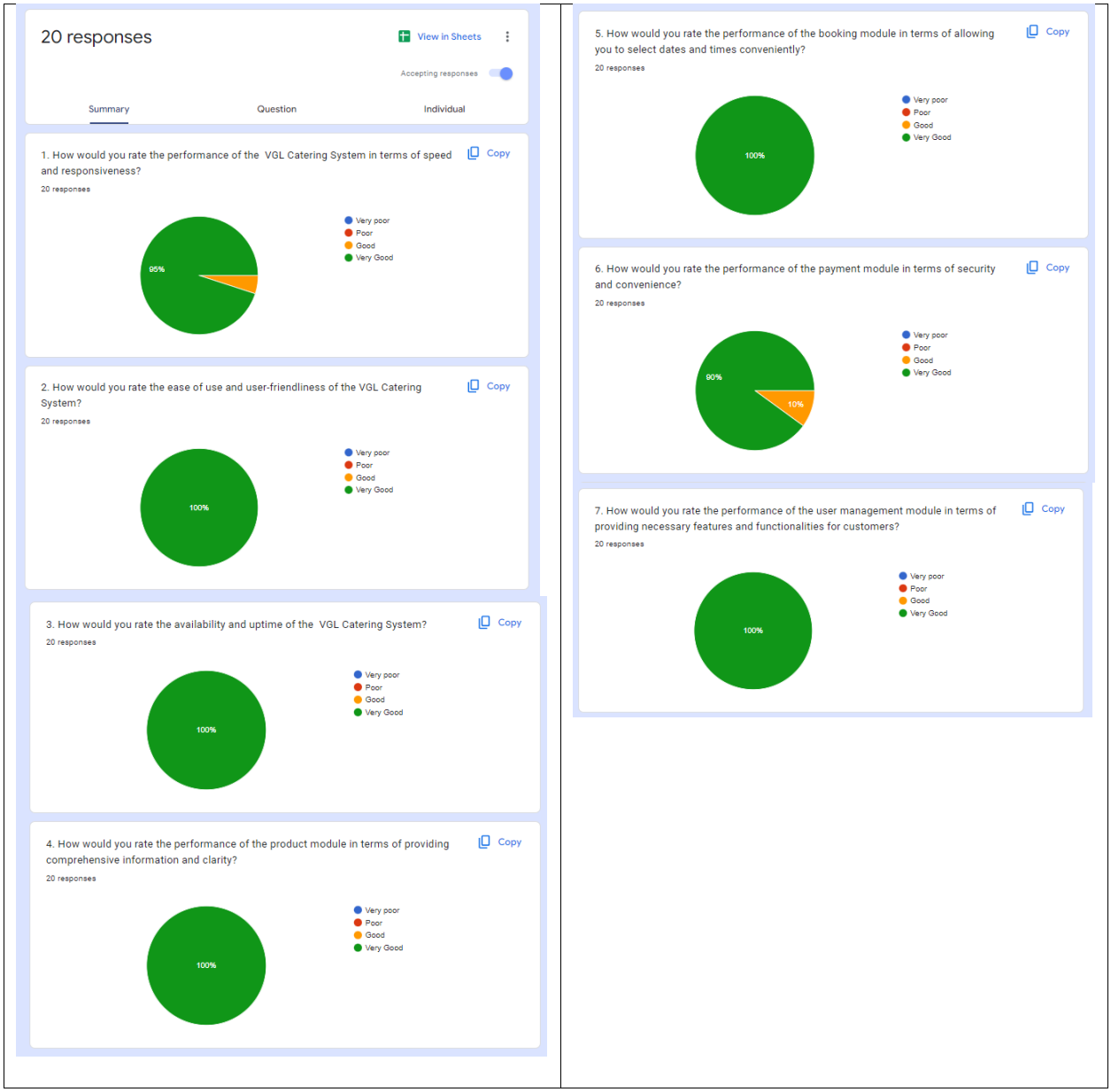


Figure 27: Result of User Acceptance test

5. Conclusion

In conclusion, the development of the VGL Cate, the web-based Catering Booking System for VGL Catering has successfully addressed the challenges faced by the organization in managing their booking processes. The project has resulted in a user-friendly web-based system that streamlines the booking system, facilitates online payments, and improves overall customer experience. Through an object-oriented approach and rigorous testing, the system has achieved its objectives of providing a structured and scalable system design, utilizing web technology and ensuring quality and reliability. Overall, this project has proven to be a successful solution that enhances efficiency, customer satisfaction, and the overall management of VGL Catering.

Acknowledgment

I would want to offer my heartfelt gratitude to everyone who made it possible for me to complete this report. First of all, I wish to express my sincere gratitude and appreciation to my supervisor, DR Rosmamalmi binti Mat Nawi for her guidance and patience in supervising the writing of this project

and the development of VGL Catering Booking System: VGL Cate using object-oriented approach. I would like to express my gratitude for her advice and encouragement. The suggestions given are helpful and serve as a source of encouragement as the project progresses.

References

[1] R. Geocke, "The Evolution of Online Booking Systems," in *Handbook of e-Tourism*, 2020, pp. 1-25.

[2] H. Huang, S. Q. Liu, J. Kandampully and M. Bujisic, "Consumer responses to scarcity appeals in online booking," *Annals of Tourism Research*, vol. 80, p. 102800, 2020.

[3] V. Chavan, P. Jadhav, S. Korade and P. Teli, "Implementing Customizable Online Food Ordering System," *International Journal of Innovative Science, Engineering & Technology*, vol. IV, pp. 722-727, 2015.

[4] J. Smith, "The impact of web-based systems on organizational efficiency," *Journal of Applied Technology*, vol. 2, no. 15, pp. 123-145, 2021.

[5] S. Johnson and M. Brown, "Web-Based Systems: Enhancing Accessibility and User Experience," *Journal of Information Technology*, vol. III, no. 25, pp. 45-67, 2020.

[6] H. Lee and S. Kim, "Cost-Effectiveness Analysis of Web-Based Systems in Small Businesses," *Small Business Economics*, vol. 4, pp. 789-807, 2019.

[7] L. Singh, "On The Go: Restaurant Management System," *International Journal for Research in Applied Science and Engineering Technology*, vol. 5, pp. 507-510, 2017.

[8] W. W. Royce, "Managing the development of large software systems: concepts and techniques," *Proceedings of the 9th international conference on Software Engineering*, pp. 328-338, 1987.

[9] M. Martin, "Functional Requirements vs Non-Functional Requirements: Differences," 2021.