TRƯỜNG ĐẠI HỌC FPT

MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

Service Link System to enhance operations in Pizza restaurant combined with workshop model

Group GSP25SE12	
Group members	Trương Sỹ Quảng – SE160326 Vũ Nhật Hào – SE161380 Nguyễn Hoàng Bảo – SE172266 Nguyễn Hưng Hảo – SE172788 Tống Trường Thanh – SE160320
Supervisor	Nguyễn Nguyên Bình
Project code	SP25SE030

- Ho Chi Minh City, May 2025 -

Table of Contents

List of Tables

List of Figures

Acknowledgements

We would like to express our heartfelt gratitude to our thesis advisor, Mr. Nguyễn Nguyên Bình, for his exceptional guidance, steadfast support, and thoughtful feedback throughout the entire process of our research and thesis writing. His expertise and encouragement have been pivotal in shaping the direction and quality of our work.

We are sincerely thankful to the members of the thesis review committee for their valuable time and effort in thoroughly reviewing our drafts and offering constructive critiques. Their insights have greatly contributed to the refinement and enhancement of this thesis.

Our sincere appreciation is extended to Ms. Nguyễn Thị Cẩm Hương, Ms. Vũ Thị Thanh Vân, Mr. Lâm Hữu Khánh Phương, and Mr. Đỗ Tấn Nhàn for being our reviewers. Their thorough evaluation, insightful feedback, and constructive suggestions have been invaluable in strengthening the academic quality of our research. We are deeply grateful for their time, dedication, and expertise, which have significantly contributed to the success of our study.

We are immensely grateful to our families for their constant support, patience, and motivation throughout our academic journey. Their love has been a source of strength and inspiration.

Special thanks go to our friends, whose encouragement, thoughtful discussions, and collaboration have enriched our research experience and made the journey more rewarding.

Finally, we extend our appreciation to all individuals and organizations who have offered their assistance, resources, or support during the course of this research. Your contributions have been truly invaluable.

To everyone who has supported us in this academic endeavor — thank you. Your presence and help have been instrumental to the successful completion of this thesis.

Definition and Acronyms

Acronym	Definition
BR	Business Rule
ERD	Entity Relationship Diagram
GUI	Graphical User Interface
UC	Use Case
SRS	Software Requirements Specification
API	Application Programing Interface
Opslink	Service Link System to enhance operations in Pizza restaurant
Manager	Users who use web applications to manage subcategory, warehouses, export, import,...
Staff	Users who uses web application to manage inventory
Chef	Users who uses web application to request the export order

Table 1: Definition and Acronym

I. Project Introduction

1. Overview

1.1 Project Information

- Project name: Service Link System to enhance operations in Pizza restaurant
- Project code: GSP25SE12
- Group name: SP25SE030
- Software type: Web application & Mobile application

1.2 Project Team

1.2.1 Supervisor

Full Name	Email	Phone Number	Title
Nguyễn Nguyên Bình	BinhNN7@fe.edu.vn	0947 430 777	Lecturer

Table 2. Supervisor

1.2.2 Team member

Full Name	Email	Mobile	Role
Trương Sỹ Quảng	quangtsse160326@fpt.edu.vn	0888509299	Frontend
Vũ Nhật Hào	haovnse161380@fpt.edu.vn	0379231040	Backend
Nguyễn Hoàng Bảo	baonhse172266@fpt.edu.vn	0934140524	Backend
Nguyễn Hưng Hảo	haonhse172788@fpt.edu.vn	0946000406	Backend
Tống Trường Thanh	thanhttse160320@fpt.edu.vn	0967992202	Frontend

Table 3. Team member

2. Product Background

In the competitive restaurant industry, efficiency and customer satisfaction are paramount. Traditional methods of managing table orders and waitstaff tasks can be prone to errors, delays,

and miscommunications. These inefficiencies can lead to longer wait times, incorrect orders, and decreased customer satisfaction. The problems may include:

Order Accuracy: Manual order taking is susceptible to mistakes, especially during peak hours, leading to incorrect or incomplete orders being served to customers.

Wait Times: Delays in order processing and delivery can result in long wait times, causing customer frustration and reducing table turnover rates.

Communication Gaps: Miscommunication between waitstaff and kitchen staff can lead to errors in order preparation and delivery, further compounding inefficiencies.

Staff Coordination: Inefficient task assignment and tracking for waitstaff can result in uneven workload distribution, missed tasks, and decreased staff productivity.

Customer Experience: A lack of real-time feedback and updates on order status can leave customers in the dark, impacting their overall dining experience.

The proposed solution aims to address these issues by developing a smart table-ordering and waitstaff management system. This system leverages modern technology to automate and streamline the process of taking orders, communicating with the kitchen, and managing waitstaff tasks from order preparation to delivery. By integrating a user-friendly interface for customers, real-time order tracking for kitchen staff, and efficient task management for waitstaff, the system seeks to enhance operational efficiency, reduce errors, and significantly improve the customer dining experience.

3. Existing Systems

3.1 App 1: DominoPizza(https://dominos.vn/)

- **Pros**:
    - Features real-time location tracking and address verification
    - Offers multilingual support for diverse customers
    - Provides comprehensive order and delivery status monitoring
- **Cons:**
    - Mandatory user authentication for order placement
    - No visibility into back-of-house kitchen operations
    - Does not support table reservation and interactive workshop booking.

4. Business Opportunity

In the restaurant industry, a smart restaurant management application addresses operational inefficiencies through integrated digital solutions. The system enables customers to access digital menus via QR codes, place customized orders through mobile devices, monitor order status in real-time, and submit immediate feedback, thereby optimizing order accuracy and service delivery.

The platform implements automated task assignment protocols for service staff, utilizing real-time notifications to systematically distribute workload and increase operational productivity while reducing service delays. Management interfaces incorporate comprehensive analytics tools that monitor staff performance metrics, order processing efficiency, and customer satisfaction indicators, facilitating data-driven operational decisions.

Through targeted solutions addressing industry-specific challenges, the platform demonstrates substantial potential for operational optimization, customer retention improvement, and profitability enhancement in the restaurant sector.

5. Software Product Vision

In response to evolving urban dining preferences that blend technological convenience with personalized service, OpsLink envisions becoming the practical solution for efficient and reliable restaurant management in premium casual dining establishments. Designed specifically for restaurants serving urban professionals, family groups, and social gatherings, OpsLink aims to enhance the dining experience while maintaining the essential human touch that defines quality service.

The vision includes:

To empower restaurant owners and employees with comprehensive digital tools that optimize workflows, enhance service quality, and increase profitability through automated processes and real-time monitoring.

To enhance the dining experience through advanced digital solutions. Customers can effortlessly scan QR codes to access interactive digital menus, place and customize orders using their mobile devices, track their order status in real-time, and provide instant feedback, ensuring a smooth and enjoyable dining experience.

To optimize staff productivity and satisfaction through automated task assignments, real-time notifications, and efficient workload management, ensuring timely and accurate service.

To deliver actionable insights through advanced analytics and reporting tools, facilitating informed decision-making on staff performance, order processing, and customer satisfaction..

To drive customer engagement and encourage ongoing patronage through digital voucher management, supported by an intuitive and user-friendly interface.

To create a flexible dining environment that accommodates diverse customer preferences, from fully digital self-service to staff-assisted experiences, ensuring both individual diners and large groups can enjoy seamless service.

To support sophisticated dining establishments in maintaining their premium service standards while leveraging digital efficiency, particularly for restaurants serving quality-conscious urban customers who expect both technological convenience and personal attention.

OpsLink is a management platform designed for premium casual dining, optimizing operations while maintaining personalized service. It helps restaurants balance technology with hospitality, enhancing efficiency and customer experience in urban markets where digital convenience meets social dining expectations.

6. Project Scope and Exclusions

6.1 Major Features

6.1.1 Manager use web application

As a OpsLink Manager, I want to view report and manage system with the following features:

FE-01. Manage users' account: create staff accounts,view users' profile, update staff accounts

FE-02. Able to track system's all activities (which activity done by who and when)

6.1.2 Manager use web application

As a user who uses manager OpsLink web application, I want to use these following features:

FE-03. Manage staff account: view profile, change password

FE-04. Manage menu item: create menu item, view menu item details, update menu item, add menu item to workshop

FE-05. Manage zone: create zone, view zone details, update zone

FE-06. Manage table: create table, view table details, update table

FE-07. Manage customer's restaurant request: view request details

FE-08. Manage order: view order details, update order item status

FE-09. Manage voucher: create vouchers in batches, view voucher batch details, update voucher batch

FE-10. Manage receipt: create receipt, view receipt details, update receipt

FE-11. Manage swap working slot request: view swap working slot request details, update swap working slot request, approve swap working slot request, deny swap working slot request

FE-12. Manage reservation: view reservation details

FE-13. Manage shift: create shift, view shift details, update shift, assign staff with shift, approve working slot registration

FE-14. Manage food options: create food options, view food options details, update food options

FE-15. Manage workshop: view workshop, update workshop

6.1.3 Staff use web application

As a user who uses staff OpsLink web application, I want to use these following features:

FE-16. Manage working slot: view working slot details, register working slot

FE-17. Manage receipt: view receipt details

FE-18. Manage table: view table details, update table, assign customer to table, merge table, swap table for customer

FE-19. Manage swap working slot request: create request, view request details, update request

FE-20. Manage order: view order details, update order items status

FE-21. Manage staff account: view profile, change password, log in, log out

FE-22. Manage customer's restaurant request: view request details, get real-time notification

6.1.4 Responsive web application for chef

As a user who uses chef OpsLink web application, I want to use these following features:

FE-23. Manage order: view order details based on sections, update order items status

FE-24. Manage food and food options: view food and food options

FE-25. Manage chef account: view profile, change password, log in, log out

6.1.5 Responsive web application for customer

As a user who uses customer OpsLink web application, I want to use these following features:

FE-26. Manage customer's review: create review, view review, update review

FE-27. Manage customer's bill: view bill

FE-28. Manage reservation: create reservation, view reservation status, update reservation

FE-29. Manage voucher: view voucher, use voucher

FE-30. Manage order: place order, view order details, update order

FE-31. Manage food: view food details, add food to order with customized options,

FE-32. Manage payment method: use payment method

FE-33. Manage restaurant service's request: create request, view request

FE-34. Manage workshop: register workshop, view workshop details

6.2 Limitations and Exclusions

LE-01. Focus solely on one specific type of service.

LE-02. The application supports only one language (e.g., English) by default.

LE-03. The application integrates with third parties.

II. Project Management Plan

1. Overview

1.1 WBS and Estimation

#	WBS Item	Complexity	Est. Effort
1	***Initiating***	Complex	***27***
1.1.	Collect requirement	Complex	14
1.2.	Define stakeholders	Medium	7
1.3.	Define Project Objectives	Medium	6
2	***Planning***	Complex	***24***
2.1.	Define Project Scope	Complex	12
2.1.1.	Develop Work Breakdown Structure (WBS)	Medium	6
2.1.2.	Identify milestones	Medium	6
2.2.	Manage Resource and Schedule	Medium	7
2.2.1.	Create detailed Project Schedule	Medium	5
2.2.2.	Create list of tools, resources for usage	Simple	2
2.3.	Plan risks and quality requirement	Medium	5
2.3.1.	Develop a risk management plan	Medium	5
2.3.1.1	Identify risks	Simple	3
2.3.1.2	Define mitigations strategies	Simple	2
3	***Executing***	Complex	***307***
3.1.	**Preparation and Training**	Complex	**40**
3.1.1.	Plan schedule	Medium	8
3.1.2.	Workflow training for the team	Medium	4

3.1.3.	Designing Database	Complex	10
3.1.4.	Prepare knowledge in .NET Core Framework	Medium	9
3.1.5.	Prepare knowledge in React, React Native	Medium	9
3.2.	**System Design**	Complex	**29**
3.2.1.	Design Customer system	Complex	12
3.2.2.	Design Manager system	Complex	10
3.2.3.	Design Staff system	Simple	3
3.2.4.	Design Chef system	Medium	4
3.3.	**Prototyping**	Complex	**14**
3.3.1.	Mockup	Complex	14
3.4.	**Implementation**	Complex	**200**
3.4.1.	*Implement Base feature*	Medium	*7*
3.4.1.1.	Basic CRUD	Medium	7
3.4.2.	*Implement Authentication Authorization*	Medium	8
3.4.2.1	Login	Simple	3
3.4.2.1	Log out	Simple	1
3.4.2.1	Register Account	Medium	5
3.4.3.	*Implement Manager feature*	Complex	*74*
3.4.3.1	Home main	Simple	3
3.4.3.2	Profile	Simple	2
3.4.3.3	Search	Simple	2
3.4.3.4	Manage Category	Simple	2
3.4.3.5	Manage Food	Simple	3
3.4.3.6	Manage Zone	Medium	5
3.4.3.7	Manage Table	Medium	5
3.4.3.8	Manage Customer Requests	Medium	5
3.4.3.9	Manage Staff/Chef Accounts	Simple	3

3.4.3.10	Manage Vouchers	Medium	7
3.4.3.11	Manage Payment Method	Medium	4
3.4.3.12	Manage Staff Request	Complex	10
3.4.3.13	Manage Workshop Slots	Complex	12
3.4.3.14	Manage Shifts	Medium	5
3.4.3.15	Manage Recipes and Food Options	Medium	6
3.4.4.	*Implement Staff feature*	Complex	*39*
3.4.4.1	Home main	Simple	3
3.4.4.2	Profile	Simple	2
3.4.4.3	Search	Simple	2
3.4.4.4	Manage Shifts	Medium	7
3.4.4.5	Manage Staff Request	Complex	10
3.4.4.6	Manage Zone	Medium	5
3.4.4.7	Manage Receipts	Medium	5
3.4.4.8	Manage Tables	Medium	5
3.4.5.	*Implement Chef feature*	Complex	*24*
3.4.5.1	Home main	Simple	3
3.4.5.2	Profile	Simple	2
3.4.5.3	Search	Simple	2
3.4.5.4	Manage Staff Request	Complex	10
3.4.5.5	Manage Orders	Medium	7
3.4.6.	*Implement Customer feature*	Complex	*48*
3.4.6.1	Home main	Simple	3
3.4.6.2	Profile	Simple	2
3.4.6.3	Search	Simple	2
3.4.6.4	Manage Customer Requests	Medium	5
3.4.6.5	Manage Customer Reviews	Simple	2

3.4.6.6	Manage Customer Vouchers	Simple	2
3.4.6.7	Manage Orders	Complex	12
3.4.6.8	Manage Reservation	Complex	14
3.4.6.9	Manage Workshop Registration	Medium	6
3.5.	**Testing**	Complex	**24**
3.5.1.	Unit Test	Medium	8
3.5.2.	Integration Test	Medium	8
3.5.3.	System Test	Medium	8
4	***Monitoring and Controlling***	Complex	***34***
4.1.	Control the process	Complex	17
4.2.	Track performance and quality	Complex	17
5	***Closing***	Complex	***26***
5.1.	Review	Complex	26
5.1.1	System Manageristrator Interface Review	Medium	4
5.1.1	Customer Interface Review	Medium	6
5.1.1	Employee Interface Review	Medium	5
5.1.1	Coding Review	Medium	5
5.1.1	Documentation Review	Medium	6
Total Estimated Effort (man-days): 418 days			

Table 4: WBS and Estimation

1.2 Project Objectives

#	Quality Stage	No. of Defects	% of Defect	Notes
1	Unit Test	12	20	
2	Integration Test	15	20	
2	System Test	42	70	

3	User Acceptance Test	6	10	

Table 5: Project Objectives

1.3 Project Risks

#	Risk Description	Impact	Possibility	Response Plans
1	Limited Expertise in New Technologies	High	High	Allocate training time and provide learning resources early in the project.
2	Difficulties in System Integration and Testing	Medium	Medium	Implement phased testing, use test data, and involve users in validation.
3	Team Collaboration and Communication Issues	High	High	Establish clear communication protocols, use project management tools, and schedule regular check-ins.

Table 6: Project Risks

2. Management Approach

2.1 Project Process

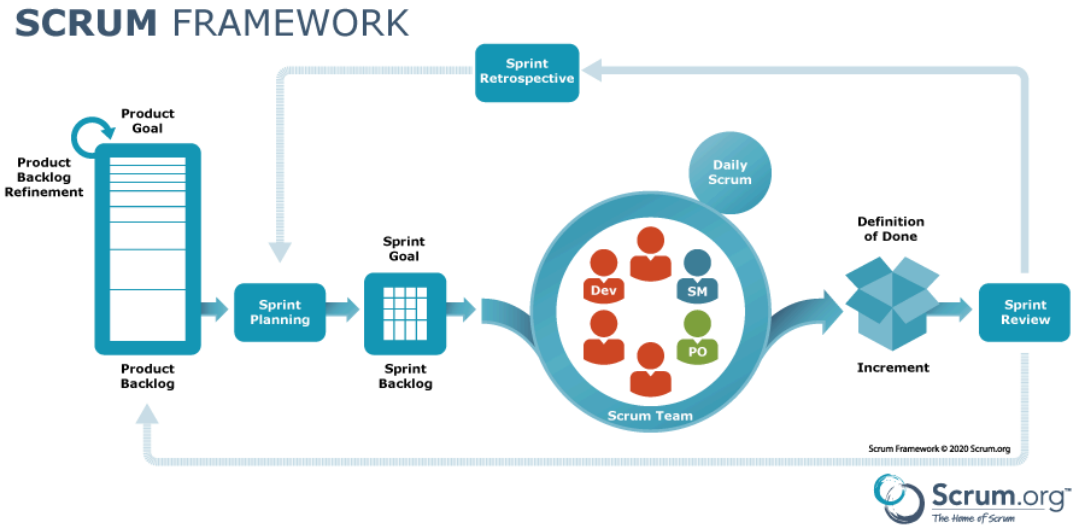


Figure 1: Scrum framework

Reference: https://www.scrum.org/resources/what-is-scrum

This project is developed using Scrum model – part of an Agile framework for project development because of the following reasons:

- Release and review features weekly.
- The project duration is about 16 weeks.
- Flexible to requirement changes.

2.2 Quality Management

In this project, to improve the project quality, we use the following approach:

- API convention:
  - Name the API Route that maps to the entity in the system
  - Each API response must return the following general information:
    - Status code: 200, 204, 400, 401, 404, …
    - Message return
    - Data: an array that contains all data items
- Coding convention:
  - Set the meaningful names for variables
  - Add comments to code can be confusing
  - Use Pascal casing for naming Enum, Class and Interface
- Unit Testing: Each member has the responsibility to write a unit test to ensure a function/method works appropriately.
- Integration Testing: After finishing coding one module/screen both members working on the client-side and server-side need to execute integration testing to ensure the module/screen can work smoothly from client-side to server-side.
- System Testing: After finishing coding for whole systems, all team members need to re-execute all test cases and free tests to ensure the system is working stable.

2.3 Training Plan

Training Area	Participants	When, Duration	Waiver Criteria
.NET Core Framework	Vu Nhat Hao Nguyen Hoang Bao Nguyen Hung Hao	Week 1 – 5 days	Mandatory
Azure, Firebase	Vu Nhat Hao Nguyen Hoang Bao Nguyen Hung Hao	Week 1 – 5 days	Mandatory
GitHub	All member	Week 1 – 3 days	Mandatory

Table 7: Training plan

3. Project Deliverables

Sprint #	Sprint Objective	Duration	Deliverable
1	Project Plan Document, SRS Document	14 days	Project Planning, Software Requirement
2	Design Document	14 days	Architecture Design, Basic Design, Detail Design and Database Design
3	Authentication Module, User Module, Role Module, Permission Module	14 days	Code & System test cases
4	Product Module, Schedule Shift Module, Zone Table Module	14 days	Code & System test cases
5	Workshop Module, Reservation Module	14 days	Code & System test cases

Table 8: Project deliverables

4.Responsibility Assignments

Responsibility	Trương Sỹ Quảng	Vũ Nhật Hào	Nguyễn Hoàng Bảo	Nguyễn Hưng Hảo	Tống Trường Thanh
Backend for system	I	A	R	R	I
Front end web	A	I	I	I	R
Server	I	I	I	A	I
Project Planning & Tracking	A	C	I	I	I
Prepare Project Introduction Document	I	C	R	A	R
Prepare SRS Document	R	A	R	D	C
Prepare SDD Document	A	C	C	D	R
Prepare SIT Document	I	R	R	A	C
Prepare System User's Manual Document	A	C	I	D	R

Table 9: Responsibility Assignments

5. Project Communication

5.1 Communication

Communication Item	Who/Target	Purpose	When, Frequency	Type, Tool, Method
Working with advisor	Supervisor and team members	● Review documentation ● Demonstrate features ● Evaluate progress and result	2 times per week	Google Meet, Zalo

Working in team	Team member	● Raise opinions, members problems ● Ask for help from other members ● Report working status to leader	Always	Google Meet, Messenger, Face to face

Table 10: Communication Plan

5.2 External Interface

5.2.1 FPTU Contacts

Function	Contact Person (name, position)	Contact address (email, telephone)	Responsibility
Supervisor	Nguyễn Nguyên Bình	BinhNN7@fe.edu.vn	● Provide documents template. ● Guide project team with problem ● Review project result ● Supervise project status

Table 11: FPTU Contacts

6. Configuration Management

6.1 Document Management

Due to real-time data synchronization, we opt for Google Drive as our document management platform. Google Drive facilitates seamless real-time collaboration among team members. We generate documents on Google Drive using templates and then allocate document-writing tasks to members based on their predefined responsibilities. Each member has instant access to view the outcomes of the work contributed by others.

6.2 Source Code Management

We opt for GitHub to manage our source code. The version control system enables team members to efficiently, conveniently, and easily collaborate on source code, especially when resolving conflicts. It proves particularly helpful for retrospection, allowing for a quick identification of changes made by collaborators over time.

6.3 Tools and Infrastructures

Programming Languages	Java, JavaScript
Framework	Spring Boot, ReactJS, React Native
DBMS	SQL Server
IDEs/Editors	Visual Studio Code, Visual Studio 2020
UML tools	Draw.io
Version Control	GitHub
Deployment server	Vercel, Self-hosted Virtual Machine
Project Management Tool	Trello

Table 12: Tool and Infrastructures

III. Software Requirements Specification

1. Product Overview

1.1 Manager Requirements

For Managers, they have the authority to manage accounts of staffs and chefs. Additionally, Managers can access reports related to orders and order items and other activities recorded on the system. This capability enables Manageristrators to ensure effective customer optimization, enhanced service quality, and make decisions based on detailed item information.

1.2 Manager Requirements

For Managers, they have the authority to manage comprehensive aspects of restaurant operations. This system provides managers with extensive capabilities to oversee staff, menu, operational zones, and critical business processes. The platform enables managers to efficiently handle staff accounts, menu configurations, table management, customer requests, orders, vouchers, receipts, staff request, bookings, shifts, and food options. This capability empowers managers to ensure effective operational control, optimize restaurant workflows, and make data-driven decisions that enhance overall business performance.

1.3 Staff Requirements

For Staff members, the OpsLink provides a comprehensive set of tools designed to support their daily operational needs and workplace interactions. The system offers intuitive features that enable staff to manage their professional activities, including personal account management, shift tracking, staff request creation, order handling, and access to essential restaurant information. This digital platform empowers staff to efficiently navigate their work responsibilities, communicate their needs, and maintain seamless operational workflow within the restaurant

1.4 Chef Requirements

For Chefs, the OpsLink web application delivers a specialized digital platform designed to streamline culinary operations and support their daily workflow. The system provides comprehensive tools that enable chefs to manage critical aspects of their work, order management, food and menu options tracking, This digital solution empowers chefs to efficiently handle their professional responsibilities, communicate operational needs, and maintain seamless integration within the restaurant's ecosystem.

1.5 Customer Requirements

For Customers, the OpsLink web application provides a comprehensive digital platform designed to enhance dining experiences and streamline restaurant interactions. The system offers an intuitive suite of features that enable customers to manage their restaurant engagement, from service requests and bookings to orders, payments, and personal account management. This digital solution empowers customers with flexible tools to interact with the restaurant, customize their dining experience, and efficiently handle various aspects of their restaurant visit.

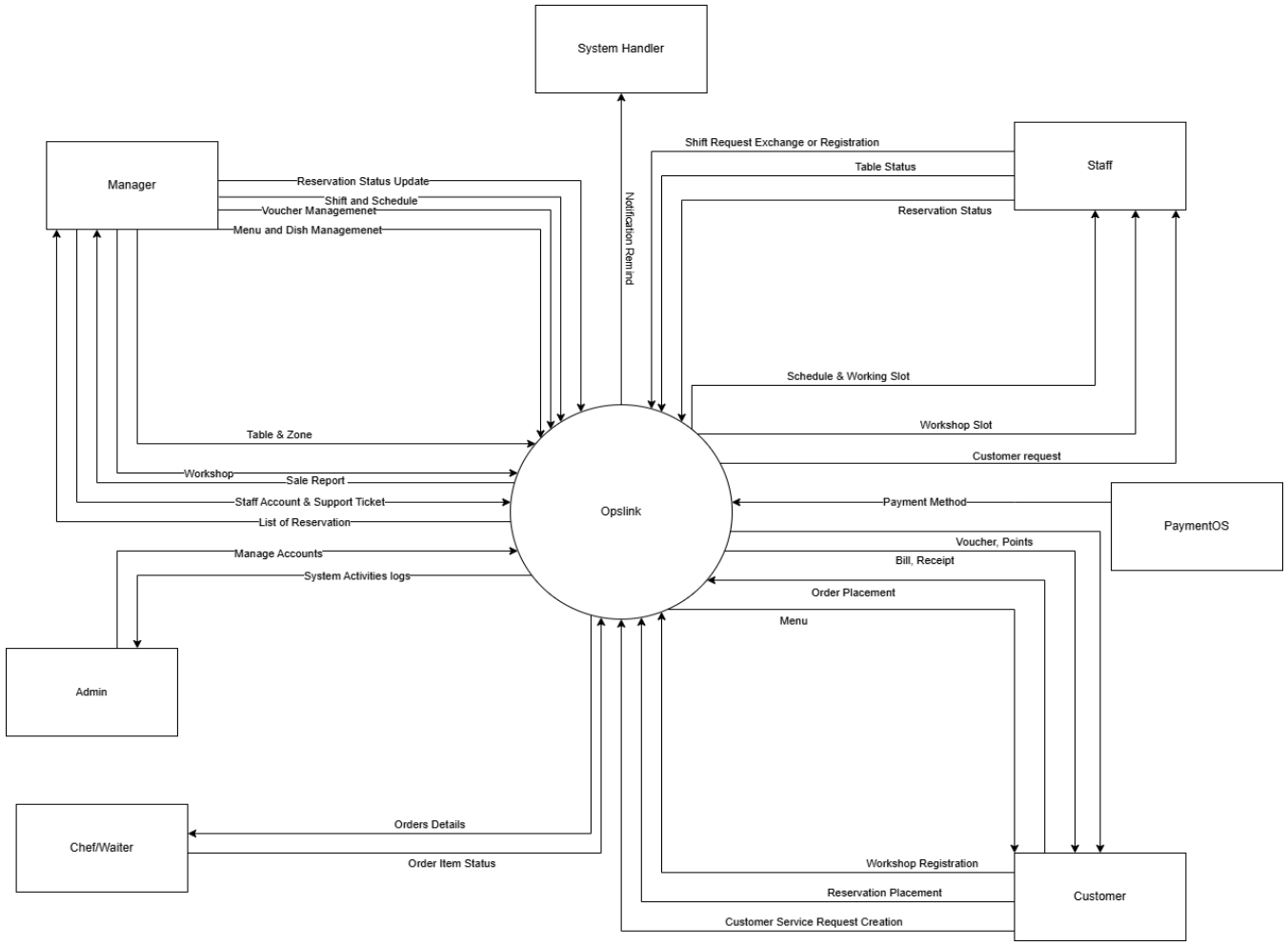


Figure 2 - Context diagram

2. User Requirements

2.1 Actors

#	Actor	Description
1	Manager	Users who use web applications to manage categories, working schedule, food and food options, workshop,...
2	Staff	Users who use web applications to view and track order, reservation, workshop registration,...
3	Chef Screen	Users who uses web application to track and update order
4	Waiter Screen	Users who uses web application to track and update order
5	Customer	Users who uses web application to manage food ordering, workshop registering, vouchers,...

Table 13 - Actors list and description

2.2 Use Cases

2.2.1 Diagrams

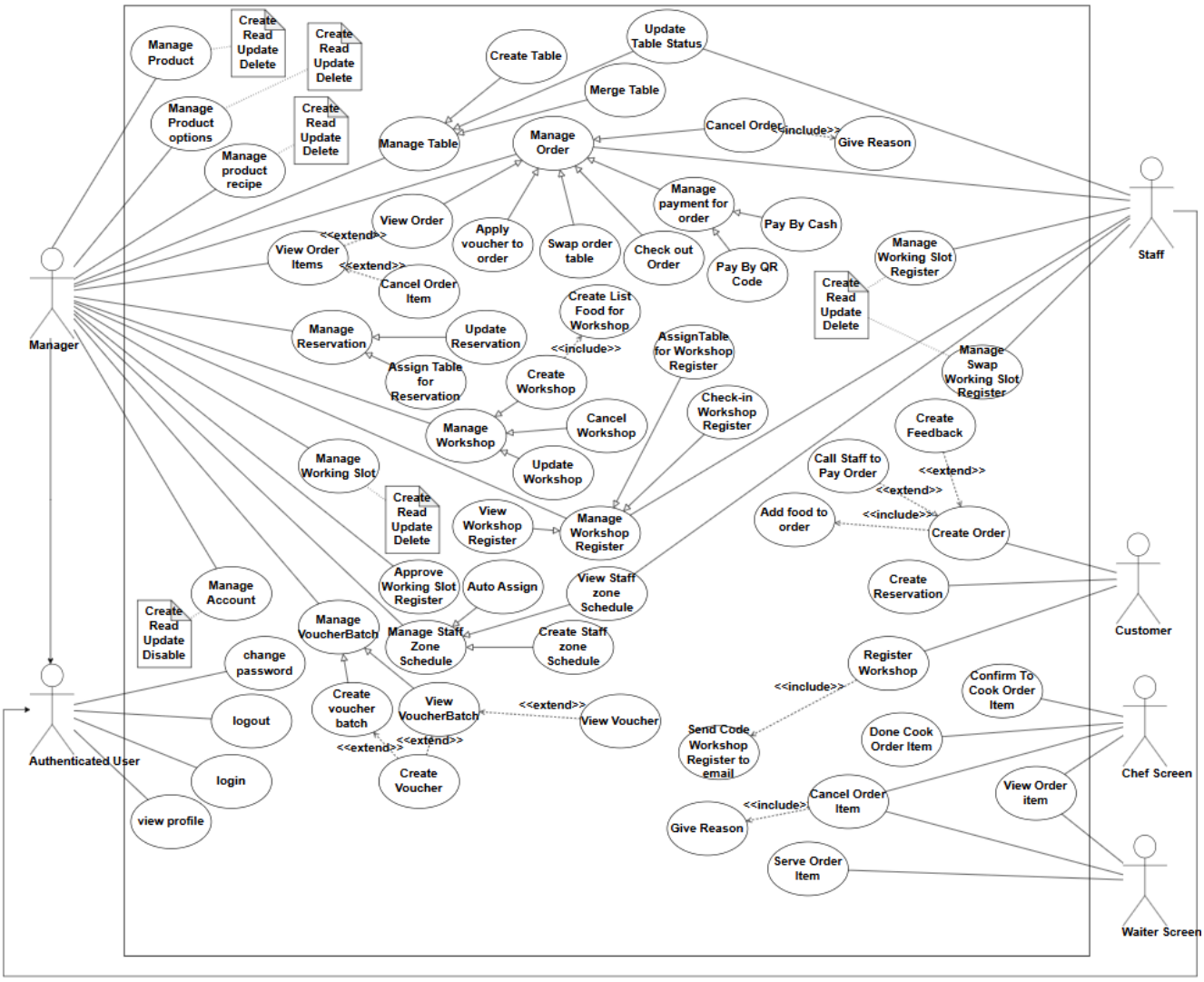


Figure 3 - Use Case diagram

2.2.2 Use Case List with description

ID	Use Case	Actors	Description
UC-01	Login	Manager , Staff, Chef	Unauthenticated user login to web application to verify role, and give them access to more features in application.
UC-02	Logout	Manager , Staff, Chef	This feature allows all users to logout the web application.

UC-03	View Profile	Manager , Staff, Chef	This feature allows all users view their profile
UC-04	Change Password	Manager , Staff, Chef	This feature allows all users change their password
UC-05	Update account	Manager , Staff, Chef	This feature allows user update account
UC-06	Create account	Manager	This feature allows user create account
UC-07	Get list of staff	Manager	This feature allows Manager view list of staff
UC-08	Update user	Manager,Staff,Chef	This feature allows Manager update information for users
UC-09	Create category	Manager	This feature allows manager create a category
UC-10	Update category	Manager	This feature allows manager update category details
UC-11	View List of category	Manager	This feature allows manager view list of category
UC-12	Get category details	Manager	This feature allows manager view information of category
UC-13	Create food	Manager	This feature allows manager create a dish
UC-14	Update food	Manager	This feature allows manager update dish
UC-15	Get list of food	Manager, Staff, Customer	This feature allows manager view list of dish
UC-16	Get food detail	Manager, Staff, Customer	This feature allows manager view information of dish
UC-17	Create food option	Manager	This feature allows manager create food option
UC-18	Get list of food option	Manager, Staff, Customer	This feature allows user view list of food option
UC-19	Get food option details	Manager, Staff, Customer	This feature allows user view information of food option
UC-20	Update food option	Manager	This feature allows user update food option
UC-21	Update food option status	Manager, Staff	This feature allows user update food option status
UC-22	Create zone	Manager	This feature allows user create zone

UC-23	Assign staff to zone	Manager	This feature allows user assign staff to zone
UC-24	Reassign Staff to Zone	Manager	This feature allows user reassign staff to zone
UC-25	Assign table for a reserved customer	Manager,	This feature allows user assign table to a customer
UC-26	Cancel assign table for a reserved customer	Manager	This feature allows user cancel assign table to a customer
UC-27	Check In Table	Customer	This feature allows user check in table
UC-28	Create table	Manager	This feature allows user create table
UC-29	close table status	Manager, Staff	This feature allows user close table
UC-30	open table status	Manager, Staff	This feature allows user open table
UC-31	lock table status	Manager	This feature allows user lock table
UC-32	Merge tables	Manager	This feature allows user merge tables
UC-33	Swap tables	Manager	This feature allows user swap tables
UC-34	Create shift configuration	Manager	This feature allows user create shift configuration
UC-35	Get list shift configuration	Manager	This feature allows user view list of shift configuration
UC-36	Get list shift	Manager	This feature allows user view list of shift
UC-37	Update shift configuration	Manager	This feature allows user update shift configuration
UC-38	Update shift status	Manager	This feature allows user update status of shift
UC-39	Assign shift to full-time staff	Manager	This feature allows user assign shift for a full-time staff
UC-40	Register shift monthly	Part-time staff/chef	This feature allows user register shift monthly
UC-41	Approve shift registration	Manager	This feature allows user approve shift registration
UC-42	Reject shift registration	Part-time staff/chef	This feature allows user reject shift registration

UC-43	Assign shift to staff according to reserved workshop	Manager	This feature allows user assign shift to staff according to booked workshop
UC-44	Update shift status	Manager	This feature allows user update status of shift
UC-45	Call absence on working slot	Staff	This feature allows user call absence on working slot
UC-46	Delete Absence	Manager	This feature allows user delete absence request on working slot
UC-47	Create voucher batches	Manager	This feature allows user create voucher
UC-48	Get list of voucher batches	Manager	This feature allows user view list of voucher
UC-49	Get voucher batch detail	Manager	This feature allows user view information of voucher
UC-50	Update voucher batch	Manager	This feature allows user update voucher
UC-51	Create vouchers by batch	Manager	This feature allows user update status of voucher
UC-52	Delete Voucher	Manager	This feature allows user delete voucher
UC-53	Create swap working slot request	Staff/Chef	This feature allows user create swap working slot request
UC-54	Get list of swap working slot request	Manager, Staff, Chef	This feature allows user view list of swap working slot request
UC-55	Get swap working slot request details	Manager, Staff, Chef	This feature allows user view information of swap working slot request
UC-56	Update swap working slot request	Staff, Chef	This feature allows user update swap working slot request
UC-57	Approve swap working slot request	Manager	This feature allows user approve swap working slot request

UC-58	Reject swap working slot request	Manager	This feature allows user deny swap working slot request
UC-59	Create workshop	Manager	This feature allows user create workshop
UC-60	Get list of workshop	Manager, Customer	This feature allows user view list of workshop
UC-61	Get workshop detail	Manager,Customer	This feature allows user view information of workshop
UC-62	Register workshop	Customer	This feature allows user register workshop
UC-63	Check in workshop	Staff	This feature allows user check in workshop
UC-64	Update workshop status	Manager	This feature allows user update status workshop
UC-65	Cancel workshop	Manager	This feature allows user cancel workshop
UC-66	Create customer's request	Customer	This feature allows user create customer 's request
UC-67	Get list of customer 's request	Manager	This feature allows user view list of customer 's request
UC-68	Get customer 's request detail	Staff	This feature allows user view information of customer 's request
UC-69	Update customer's request	Customer	This feature allows user update customer 's request
UC-70	Create reservation	Manager, Customer	This feature allows user create reservation
UC-71	Get list of reservation	Manager, Staff,	This feature allows user view list of reservation
UC-72	Get reservation detail	Manager, Staff, Customer	This feature allows user view information of reservation
UC-73	Update reservation	Manager	This feature allows user update reservation
UC-74	Update reservation status	Manager	This feature allows user update status of reservation

UC-75	Confirm reservation	Manager, Staff	This feature allows user confirm reservation
UC-76	Get bill detail	Manager, Staff, Customer	This feature allows user view information of bill
UC-77	Export bill	Manager	This feature allows user export bill
UC-78	Create order	Customer	This feature allows user create order
UC-79	Update order	Customer	This feature allows users to update order
UC-80	View order list	Manager	This feature allows users to view list of order
UC-81	Get order details	Manager, Staff, Customer	This feature allows users to view information of order
UC-82	Check out order	Manager, Staff	This feature allows users to check out order
UC-83	Cancel check out order	Manager, Staff	This feature allows users to cancel check out order
UC-84	Add food to cart	Customer	This feature allows user add food to cart
UC-85	View cart details	Customer	This feature allows users to view cart details
UC-86	Create employee account	Manager	This feature allows user to create employee account
UC-87	Update employee account	Manager	This feature allows user to update employee account
UC-88	Get list of employee account	Manager	This feature allows user to view information of employee account
UC-89	Get employee account details	Manager	This feature allows user to view information of employee account
UC-90	Create payment QR Code	Manager, Staff	This feature allows user to create payment QR Code
UC-91	Cancel payment QR Code	Manager, Staff	This feature allows user to cancel payment QR Code
UC-92	View list of payment method	Manager, Customer, Staff	This feature allows user to view list of payment method
UC-93	Create Notification	Customer	This feature allows user to get notification of Customer Request

Table 14 - Use Case List with description

2.2.3 Background Jobs

- Scheduled for Auto Update Workshop Status To OpeningToRegister
- Scheduled for Auto Update Workshop Status To ClosedRegister
- Scheduled for Auto Update Workshop Status To Opening
- Scheduled for Remind Assign Table for Reservation
- Scheduled for Auto Assign Staff Zone Schedule

2.2.4 Notification Real-time

- Notify to manager/staff when customer click “call” in customer web
- Notify to manager when customer reserve
- Notify to manager when came to Reservation time

3. Functional Requirements

3.1 System Functional Overview

3.1.1 Screen Flow

3.1.1.1 Manager Web Application Screen Flow


Login Screen
Authenticate
Home Screen
Dashboard screen
Report overview
Tables screen
Table Management Screen
Table List Screen
Zone Management Screen
Zone List Screen
Table Reservation Management Screen
Reservation List Screen
Staff Zone Management Screen
Staff List Screen
Orders Management Screen
Orders List Screen
Food Management Screen
Menu Food List Screen
Ingredient List Screen
Category List Screen
Rating Managerment Screen
Rating List Screen
Staff screen
Staff List Screen
Work Schedule Management Screen
Work Schedule List by Week Screen
Work Schedule List by Month Screen
Create Shift, Working Slot Screen
Absent Working Slot Screen
Setting Shift, Working Slot Screen
Workshop Management Screen
Workshop List Screen
Workshop Detail Screen
Create, Update Workshop Screen
Promotion Management Screen
Voucher Batch Screen
Voucher Screen
Setting Management Screen
Slot work screen
Table screen
Zone screen
Menu screen
Staff Sceen

Figure 4 - <Screen flow diagram> Manager web application

3.1.1.2 <Customer> Web Application Screen Flow

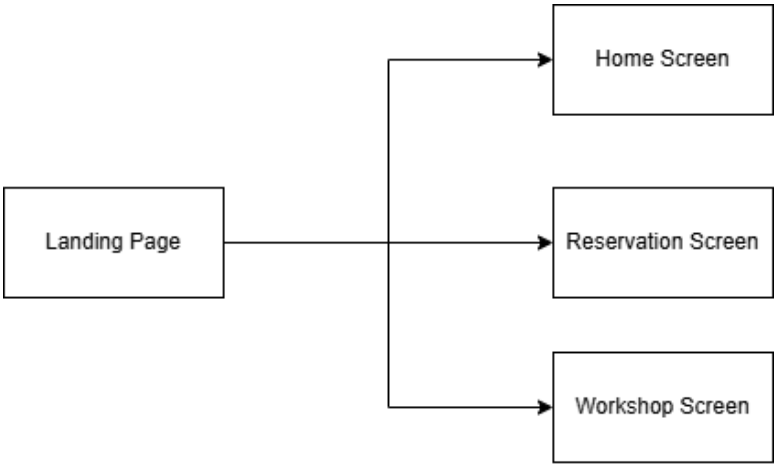


Figure 5 - <Screen flow diagram> Manager web application

3.1.1.3 <Staff> Web Application Screen Flow

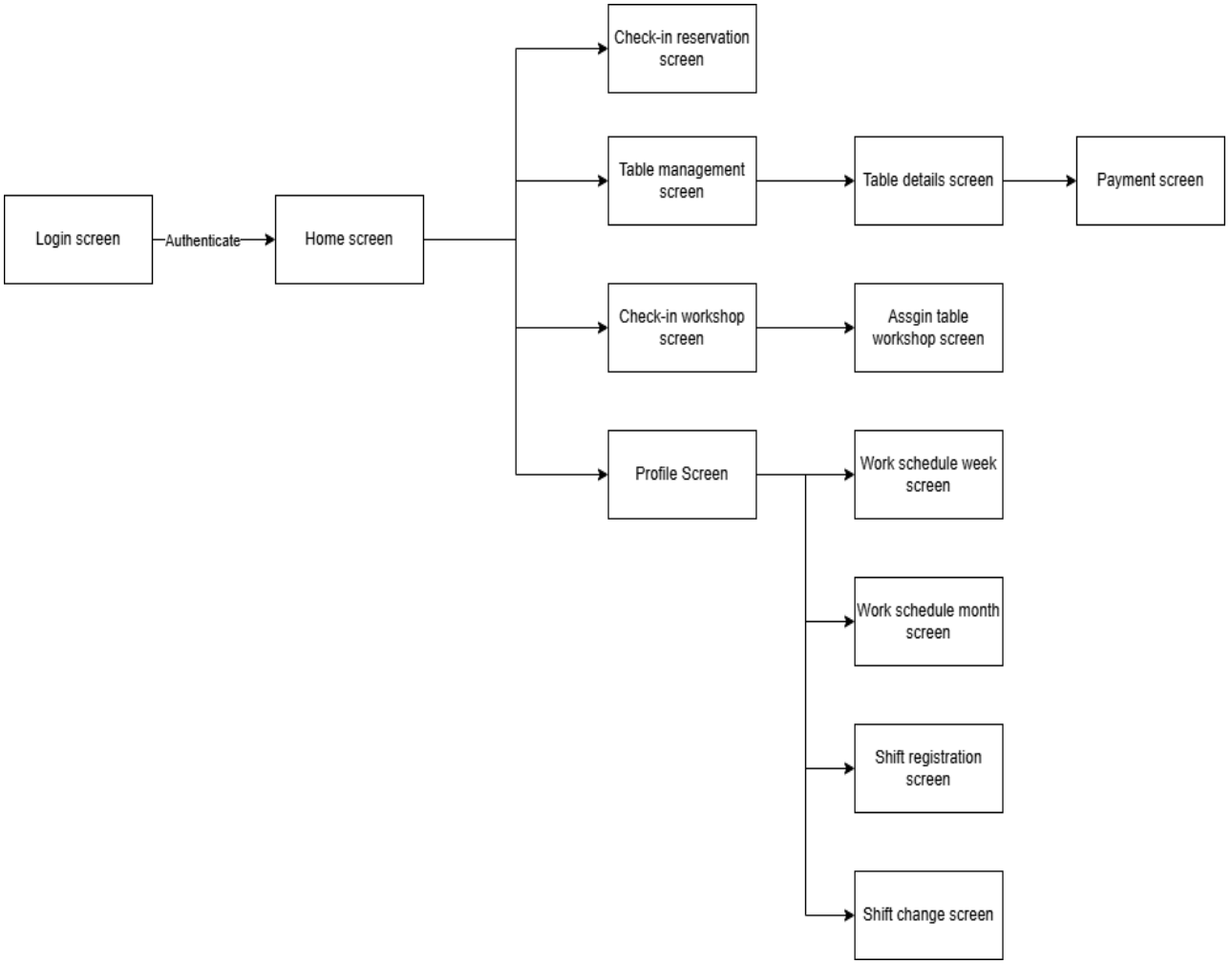


Figure 6 - <Screen flow diagram> Staff web application

3.1.1.4 <User> Web Application Screen Flow

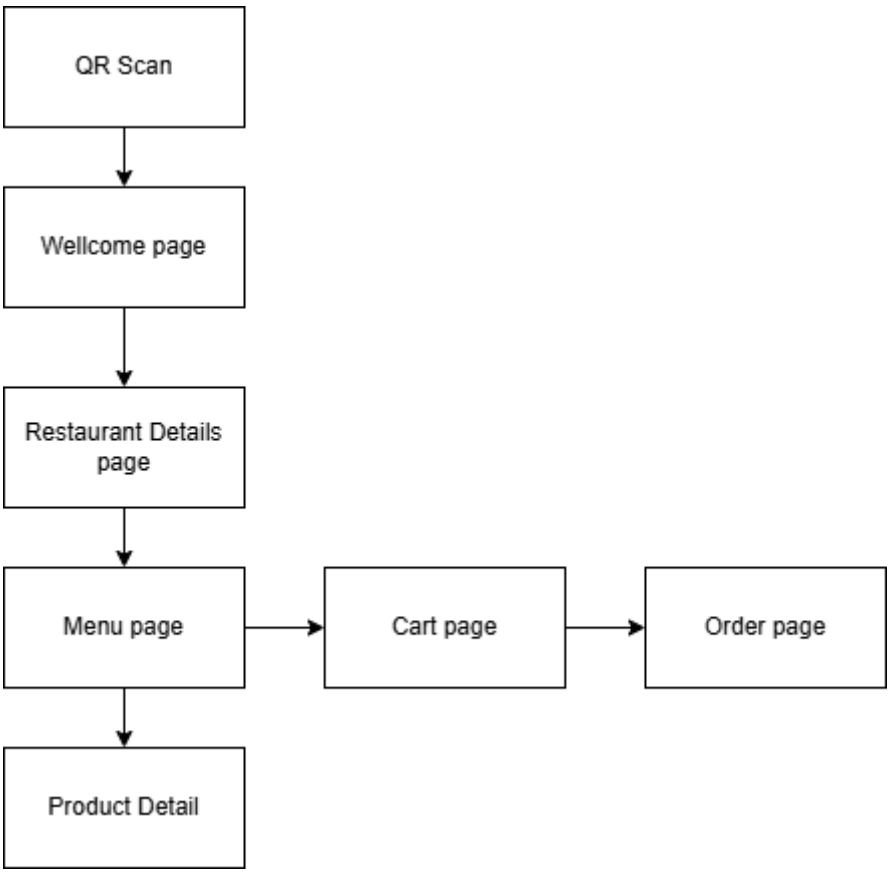


Figure 7 - <Screen flow diagram> User web application

3.1.1.5 <Chef> Web Application Screen Flow

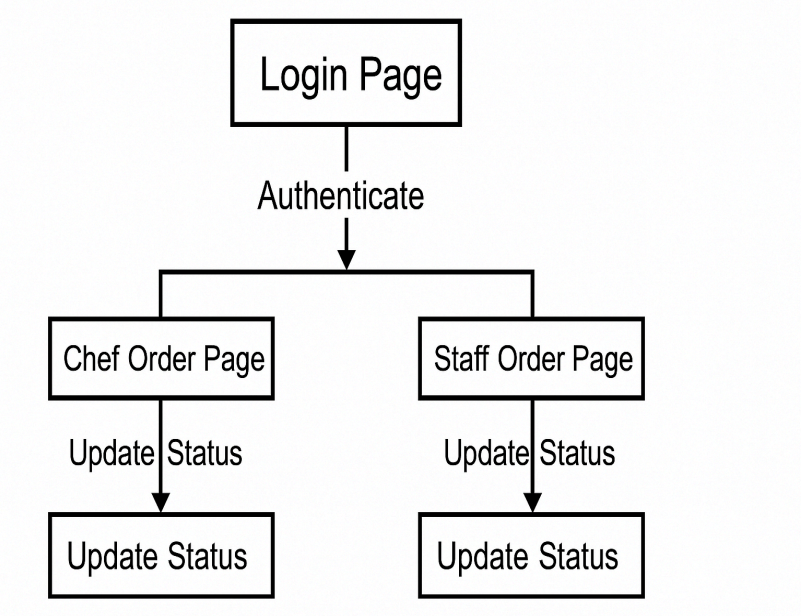


Figure 8 - <Screen flow diagram> Chef web application

3.1.2 Screen Descriptions

3.1.2.1. <Manager web> Screen description

ID	Feature	Screen	Description
1	Login	Login screen	This screen is for unauthenticated users to login. Login screen include: Manager ● Tên tài khoản: Text ● Mật khẩu: Text
2	N/A	Home Screen	- The **Home Screen** serves as the entry point to the system after a successful login. It provides a high-level overview of key business areas via a dashboard and acts as a central hub for navigating to different functional modules via the sidebar.
3	Navigate to: “Table Management”, “Zone Management”, “Table Reservation Management”	Tables screen	- Acts as the entry point to all table-related management modules.
4	- Add/Edit/Delete Tables - Assign table to zone - View active/inactive tables	Table Management Screen	- Central module for managing dining tables (CRUD operations).
5	- **Get List** of tables via API (e.g., GET `/tables`). - Filter by zone or status - Search by table name/code	Table List Screen	- Displays all tables in the system with status, zone, and capacity.
6	- Create/Edit/Delete zones	Zone management	- Manages different dining zones/areas in the restaurant.

7	- Get List of zones (e.g., `GET /zones`) - Display zone name, number of tables	Zone List Screen	- Lists all zones configured in the system.
8	- Create/Edit/Cancel reservations	Table Reservation Management Screen	- Module to manage advance table reservations made by customers.
9	- Get List of reservations (`GET /reservations`) - Filter by date, customer, table - Check reservation status (confirmed, canceled, done)	Reservation List Screen	- Displays upcoming and historical table reservations.
10-	- Create/Edit/Delete staff zones - Assign staff to zone	Staff Zone Management Screen	- Manages the assignment of staff to specific zones within the restaurant.
11	- **Get List** of staff (GET /staff) - Search staff by name or ID	Staff List Screen	- Displays a list of all active and inactive staff in the system.
12	- View order status	Orders Management Screen	- Centralized module for handling all customer orders placed in the system.
13	- **Get List** of orders (`GET/orders`) - Filter by status, table, or date - View total amount and payment status	Orders List Screen	- Shows list of all orders, including dine-in and take-away.
14	Navigate to: ● Menu Food List ● Ingredient List ● Category List	Food Management Screen	- Core module for managing menu items, categories, and ingredients.

15	- **Get List** of food/menu items - Filter by category, kitchen type (hot/cold) - Edit food info, price, availability	Menu Food List Screen	- Displays all food items available on the menu.
16	- **Get List** of ingredients (GET /ingredients) - Add/Edit/Delete ingredients	Ingredient List Screen	- Lists all ingredients used in the restaurant's menu items.
17	- **Get List** of categories (GET /categories) - Add/Edit/Delete category	Category List Screen	- Lists all food categories such as "Pizza", "Desserts", "Combos".
18	- View rating analytics and history	Rating Management Screen	- Manages customer ratings/feedback for staff, services, or orders.
19	- **Get List** of ratings - Filter by staff, order, score, or date - View rating details and feedback comments	Rating List Screen	- Displays a list of all customer-submitted ratings.
20	- Navigate to Staff List Screen	Staff Screen	- Entry point for all staff-related management.
21	- **Get List** of staff	Staff List Screen	- Lists all employees/staff members in the system.
22	- Navigate to: Weekly/Monthly schedule	Work Schedule Management Screen	- Central hub for managing all staff work shifts and schedules.
23	- Get Weekly Schedule - View shifts by day and time slot	Work Schedule List by Week Screen	- Weekly calendar view of assigned shifts per staff member.
24	- Get Monthly Schedule	Work Schedule List by Month Screen	- Monthly calendar view of all scheduled shifts.
25	- Create new shift - Set working hours, position, and zone - Assign staff to shift	Create Shift, Working Slot Screen	- Interface to create new shifts and assign staff to specific working slots.

26	- Mark absences	Absent Working Slot Screen	- Manage or report staff absences and no-shows.
27	- Configure shift durations, time blocks - Set min/max staff per slot - Define shift rules (e.g. break time, auto-assignment)	Setting Shift, Working Slot Screen	- Customize system-level settings for shifts and working slot structures.
28	- Navigate to voucher batches or individual voucher setup.	Promotion Management Screen	- Module to manage discount campaigns and vouchers.
29	- Create/edit batch - Set expiration rules, usage conditions	Voucher Batch Screen	- Manage grouped voucher campaigns (e.g., 100 vouchers for Black Friday).
30	- **Get list** of vouchers - Check usage status, customer assignment	Voucher Screen	- List of individual vouchers generated per batch.
31	- **Get list** of workshops - Filter by status, date, trainer	Workshop List Screen	- Overview of all workshops in the system.
32	- View participants, time, staff assigned - Link to update form	Workshop Detail Screen	- Displays full information about a selected workshop.
33	- Input title, date, trainer, capacity	Create, Update Workshop Screen	- Form to create a new workshop or update an existing one.
34	- Navigate to slot, table, zone, menu, and staff configuration screens.	Setting Management Screen	- Central configuration hub for system-level entities.
35	- Define time ranges (e.g., 08:00–12:00) - Set slot availability for days or staff roles	Slot Work Screen	- Configure working slot timeframes used in shift or workshop scheduling.

36	- Add/edit/delete tables	Table Screen	- Used to configure tables in the restaurant (number, capacity, code).
37	- Create/edit zones	Zone Screen	- Manage dining or staff-serving zones (e.g., A, B, Outdoor).
38	- Add/edit/remove categories	Menu Screen	- List or configure food categories used in the menu (e.g., Combo, Pizza).
39	- View all employees - Grant roles/permissions (if applicable)	Staff Screen	- System-wide list of staff accounts and roles.

Table 15: Web Screen Descriptions

3.1.2.2. <Staff Mobile> Screen description

ID	Feature	Screen	Description
1	Authentication	Login screen	Allows staff to authenticate. Includes: • Tên đăng nhập: Text input • Mật khẩu: Text input (with show/hide toggle) • Đăng nhập: Primary button
2	Check-in Booking	Check-in Đặt Bàn Screen	Let staff verify a reservation by phone number. Includes: • Nhập số điện thoại…: Text input • Search: icon button • Result list or message “Không có đặt bàn nào được tìm thấy” when empty
3	View Table Orders	Table Details Screen	Displays all order items for a selected table. Sections: • Chưa hoàn thành (0) / Hoàn thành (5): Collapsible headers • Item cards showing name, quantity, status badge, price • Footer with Mã giảm giá + + Thêm mã button and Checkout button

4	Confirm Checkout	Checkout Confirmation Screen	After tapping Checkout, shows order summary and fees: • Item total (“Tổng tiền món”) • VAT (8%) • Grand total (“Tổng tiền”) • × Hủy Checkout & Thanh Toán actions
5	Table Assignment	Your Tables Screen	Lists tables assigned to the staff’s zone. Includes search by name, filters (Tất cả/Bàn mở/Bàn đóng), zone info with star badge, and table cards showing status, name, capacity & occupancy
6	Dashboard	Home Screen	Entry point after login. Shows greeting, current position card, quick action for Check-in Đặt Bàn, and bottom navigation bar
7	Payment Method Selection	Payment Screen	Enables choosing how to pay. Includes: • Tabs: QR Code / Tiền Mặt • Display of Tổng tiền thanh toán • Prompt “Vui lòng chọn phương thức thanh toán ở trên”
8	Security	Change Password Screen	Allows updating password. Fields: • Mật khẩu hiện tại • Mật khẩu mới • Xác nhận mật khẩu mới • Xác Nhận button

9	Profile Management	Profile Screen	Displays user profile and main functions with tiles for schedule, shift registration, shift change, monthly schedule, and account settings
10	Work Schedule	Work Schedule Screen	Shows weekly list of shifts. Includes month navigator and cards for each day with shift details and zone
11	Monthly Work Overview	Work Schedule by Month Screen	Calendar view of month with dots indicating scheduled shifts, prev/next navigation, and legend
12	Daily Shift Detail	Detail work Schedule	Shows details for a selected day's shift in a modal with shift name, time range, and zone
13	Shift Registration & Summary	Đăng ký giờ làm việc Screen	Allows registering for shifts. Includes info banner, date picker, shift list with add/remove controls, registration summary card, and confirm button
14	QR Check-in	Workshop check in Screen	Enables scanning QR codes for check-in. Includes instructions, viewfinder frame, and camera toggle button
15	Shift Change	Shift Change (multi-step)	There are 4 steps: 1. Select the day with a shift: The monthly calendar has a red dot marking the day you have a shift; the two items “Shift change request sent” and “Shift change request received” show the status. 2. Select the current shift: List of your

			shifts on the selected day 3. Select the shift you want to change: List of your colleagues' shifts on another day with area & employee information. 4. Confirm shift change (modal): Displays a summary of the current shift & the shift you want to change, notes on regulations, and a "Send shift change request" button.

3.1.2.3. <Customer Mobile Web> Screen description

ID	Feature	Screen	Description
1	Scan QR Code	Scan QR Code Screen	Open camera to scan restaurant QR code; proceed to Restaurant Details on valid scan, or Error Screen on invalid/expired code.
2	View Restaurant Info	Restaurant Details Screen	Display restaurant name, address, promo banner, greeting, table code, loyalty notice, and shortcuts to call staff or checkout.
3	Error Handling	Error Screen	Inform user table is unavailable (invalid QR or closed) and offer options to contact staff or return home.
4	Browse Menu	Menu Screen	Show menu categories with search bar; list each item with image, name, price; allow tapping to view details.
5	Review & Place Order	Order Details	Summarize cart items (options,

		Screen	quantities), display prices, and allow adjusting or submitting order.
6	Welcome Guest	Welcome Screen	Display greeting message, prompt guest for their name, with a “Tiếp tục” button to proceed to Restaurant Details.

3.1.2.4. <Customer Web Application> Screen description

ID	Feature	Screen	Description
1	Landing Experience	Landing Page Screen	Display site header (logo, main nav links, “Book a Table” CTA), full-bleed hero banner with key slogan (“Order Now”, “View Menu”, “Book a Table”), and footer..
2	Table Reservation	Reservation Screen	Show restaurant info (photo, address, phone, hours, booking policies) alongside a reservation form (name, phone + OTP, date & time picker, guest count) and a “Xác Nhận Đặt Bàn” button.
3	Workshop Catalog	Workshop List Screen	List upcoming workshops with title, date badge, thumbnail, and key details (date/time, location, fee). Each card has a “Đăng ký ngay” button

3.1.2.5. <Chef Web Application> Screen description

ID	Feature	Screen	Description
1	Authentication	Login Page Screen	Show login form (username + password); upon successful authentication, move to main screen appropriate to role (Chef or Staff).

2	Chef Order Management	Chef Order Page	Lists orders (assigned to "Kitchen"/Workshop) awaiting conversion; allows the kitchen to "Confirm Cook" and "Complete" each dish.
3	Staff Service Management	Staff Order Page	List the cooked dishes and pass them to "Staff" for serving; allow staff to "Serve" the dishes to the table.

3.1.3 Screen Authorization

3.1.3.1 <Web Module>

#	Web Module / Screen	Typical Actions (C-R-U-D / Special)	Manager	Staff / Waiter	Chef / Kitchen	Customer (guest)
1	Authentication (login / logout / change PW)	C-R-U	✓	✓	✓	
2	Home Dashboard	R	✓			
3	User Account Management	C-R-U-D	✓	✓	✓	
4	Own Profile	R-U	✓	✓	✓	
5	Category Management	C-R-U-D	✓			
6	Product / Menu Management	C-R-U-D	✓			
7	Ingredient Management	C-R-U-D	✓			
8	Zone Management	C-R-U-D	✓			

9	Table Management (create, open/close, lock, merge, swap)	C-R-U-D	✓	✓		
10	Reservation Management	C-R-U-D	✓	✓		C
11	Order Management(front-of-house)	C-R-U	✓	✓		
12	Order Item Management (kitchen)	U (Confirm Cook / Done Cook)			✓	
13	Serving Management(waiter)	U (Done Serve)	✓	✓		
14	Receipt / Payment Management	C-R-U-D	✓	✓		R
15	Voucher Batch Management	C-R-U-D	✓			
16	Workshop Management	C-R-U-D	✓	✓		C
17	Working Slot & Shift Management	C-R-U-D / Approve	✓	R-C		
18	Staff Zone Schedule	C-R-U-D / Auto-Assign	✓			
19	Swap Working Slot Request	C-R-U / Approve / Deny	✓	✓		
20	Rating / Feedback	R	✓			C-R

21	Notifications	R	✓	✓	✓	✓
22	Reports (sales, activity)	R	✓	✓		

Table 16 - Screen web authorization

3.1.4 Non-Screen Functions

#	Feature	System Function	Description
1	Send notification	Send notification to user	• Sends notification to inventory staff when manager created new import request receipt • Sends notification to manager when inventory staff created new import receipt • Sends notification to inventory staff when sale staff created new customer request receipt • Sends notification to sale staff when inventory staff created new export receipt • Sends notification to inventory staff when another inventory staff created warehouse transfer
2	Send Mail	Send Mails to user	• Sends mails to users when Manager created new accounts

Table 17 - Non-Screen description

3.1.5 Entity Relationship Diagram

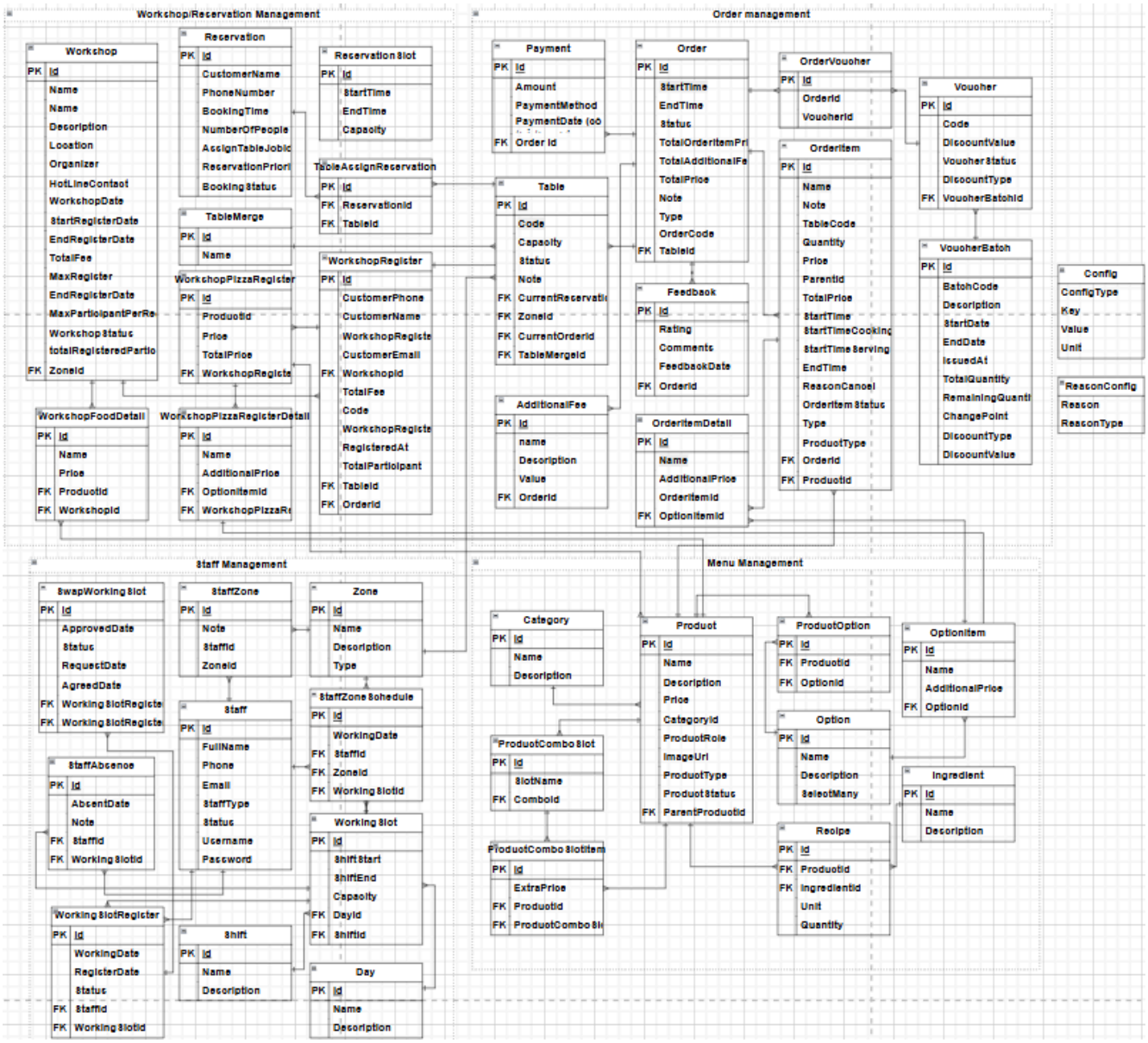


Figure 9 - ERD

Entity Description

No	Entity	Description
01	Workshop	Describe pizza workshop in the system
02	Reservation	Describe customer reservation in the system
03	ReservationSlot	Describe time slot for reservation in the system
04	Table	Describe restaurant table in the system

05	TableMerge	Describe merged tables for large group customer
06	WorkshopRegister	Describe customer registration for a workshop
07	WorkshopPizzaRegister	Describe selected pizza during workshop
08	WorkshopPizzaRegisterDetail	Describe selected pizza details during workshop (option items for each pizza)
09	WorkshopFoodDetail	Describe food items that customer can register in a workshop
10	Payment	Describe payment details for a reservation or order
11	Order	Describe food order placed by customer
12	OrderItem	Describe food item in a customer's order
13	OrderItemDetail	Describe additional details for each order item
14	OrderVoucher	Describe voucher applied to an order
15	Voucher	Describe discount voucher in the system
16	VoucherBatch	Describe batch of generated vouchers
17	Feedback	Describe feedback given by customer for an order
18	AdditionalFee	Describe additional fee applied to an order
19	Config	Describe configurable system parameters
20	ReasonConfig	Describe predefined reason codes in the system

21	SwapWorkingSlot	Describe staff working slot swap request
22	StaffZone	Describe assignment of staff to specific zones
23	Zone	Describe service zone in the restaurant
24	StaffZoneSchedule	Describe the work zone assigned to each employee based on the working day and Working slot
25	Staff	Describe restaurant staff information
26	WorkingSlot	Describe working time slot for staff
27	WorkingSlotRegister	Describe staff registration for a working slot
28	Shift	Describe shift definition in the system
29	Day	Describe day of the week
30	Category	Describe product category in the system
31	Product	Represents a product in the system, which can be either a standalone menu item (e.g., a pizza or drink) or a combo that contains multiple selection slots for choosing different items.
32	ProductOption	Describe option group available for a product
33	Option	Describe option group (e.g., topping, size)
34	OptionItem	Describe specific item within an option group
35	Ingredient	Describe ingredient used in recipes

36	Recipe	Describe ingredient composition of a product
37	ProductComboSlot	Describes selection slots within a product combo, where each slot allows choosing from a predefined list of items.
38	ProductComboSlotItem	Describes the list of products available for selection within a combo slot.
39	StaffAbsence	Describe staff absence records including staff info, working slot, and Absence Date
40	TableAssignReservation	Describe the assignment of tables to a specific reservation.

3.2 Web Application

3.2.1 Table Management Feature

3.2.1.1 Create Table

- **Function Trigger:** This function is triggered when Manager logs in to their account, clicks on the **"Bàn ăn"** button, then clicks on the **"Thêm bàn mới"** button to open the creation panel. After entering the required information, clicking the **"Thêm bàn mới"** button will create a new table.
- **Function Description:**
  - **Roles:** Manager
  - **Purpose:** To add a new table to the system with details such as table code, seat count, and area…
  - **Interface:**
    - **Mã bàn** (Required)
    - **Số chỗ** (Required)
    - **Khu vực** (Required, selection from predefined list)
  - Data processing:
    - **Input Validation**: Ensures that Table Code, Seat Count, and Zones are provided and valid.
    - **API Call**: Sends the table details to the backend for storage
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen Layout:**

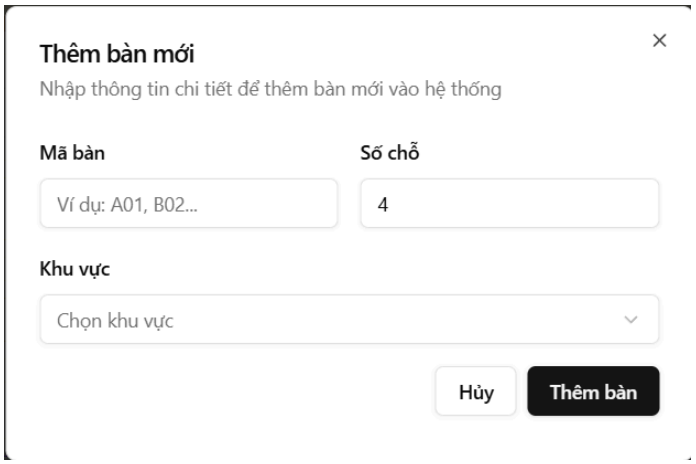


Figure 10- Create a new table

- **Function Details:**
  - Data:
    - **Mã bàn**(string, required): The unique identifier for the table (e.g., A01, B02).
    - **Số chỗ** (integer, required): The number of seats available at the table.
    - **Khu vực**(string or id, required): The selected area where the table will be located.
  - Validation:
    - **Mã bàn:**
      - Required and must not be empty.
      - Trims whitespace to ensure proper formatting.
    - **Số chỗ**:
      - Required and must be a valid number.
      - Must be greater than 0. Less than 10
    - **Khu vực**:
      - Required to select a valid area.
  - Business rules: A01, A02, A03, C01 ,C02, C03
  - Functionalities:
    - Validates all input fields upon form submission.
    - Prevents submission if required fields are empty or invalid.
    - Prevents submission if required fields are empty or invalid.
    - Sends the validated data to the backend using axios.POST
    - Handles API responses to display success or error messages.
    - Resets the form fields after a successful table creation.

3.2.1.2 Open Table

- **Function trigger:** This function is triggered when the Manager logs in to their account, navigates to the **“Bàn ăn”** section, and clicks ” **Mở bàn**" button on a table that is currently closed. Upon clicking the "Open Table" button, the table's status will be updated to "Open".
- **Function description:**
  - Roles: Manager
  - Purpose: To change the status of a closed table to an open state, making it available for new customers.
  - Interface:

        - Display current status: “**Bàn đóng**” (Table Closed)
        - Display Seat Capacity: Number of people the table can accommodate.
        - Button:
            - Open Table ("**Mở bàn**") — green color
    - Data Processing:
        - Input Validation: Ensure that the table is currently closed before allowing the status change.
        - API Call: sends a request to the backend to update the table status form “Closed” to “ Open”
        - Feedback: Displays success or error message based on the API response

- **Screen layout:**

Figure 11- Open a table

- **Function Details:**
    - Data:
        - TableId(string, required): The identifier of the table to be opened
        - New Status(string): “Open” (or “ Bàn trống “ in the UI)
    - Validation:
        - The table must currently be in a "Closed" state before being opened.
        - Prevent multiple clicks while the API request is being processed (disable button during API call).
    - Business Rules:
        - **BR-01**: Only tables with status "Closed" can be opened.
        - **BR-02**: Status change must be persisted immediately to ensure table availability is accurately reflected.
    - Functionalities:
        - Detects if the selected table is currently closed.
        - When “Mở bàn” is clicked: Calls backend API (axios.PUT) to update the table status.
        - Provides user feedback (success or error message).

3.2.1.3 Close Table

- **Function trigger:** This function is triggered when Manager logs in to their account, navigates to the Table Management section, and clicks the "Close Table" button on a table that is currently in the "Open" state.

  Upon clicking the "Close Table" button:

  - If the table has active orders, a warning message will appear
  - If there are no active orders, the table will be closed successfully.

- **Function description:**
  - Roles: Manager
  - Purpose: To close a table that is currently open, ensuring that no active orders are left unpaid.
  - Interface**:**
    - Display Current Status: "Bàn mở" (Table Open)
    - Display Seat Capacity: Number of seats (e.g., 4 người).
    - Display Order Status: If an order exists: "Đơn hàng: Đang phục vụ"
    - Button:
      - Close Table ("Đóng bàn") — red color
      - Maintenance ("Bảo trì") — optional action
  - Data processing:
    - Input Validation: Check if the table has an active order.
    - API Call:
      - If no active orders: Sends a request to the backend to update the table status to "Closed".
      - If there are active orders: No API call is made; a warning popup is displayed.
    - Feedback:
      - Displays a success message if the table is closed.
      - Displays warning message if there are unpaid orders.
- **Screen layout:**

Figure 12- Close Table

- **Function Details:**
  - Data:
    - **TableId** (string, required): The identifier of the table to be closed.
    - **OrderStatus** (string): The status of any active order linked to the table.
  - Validation:
    - Before closing:
      - Check if the table has an active order (OrderStatus == "Đang phục vụ" or equivalent).
    - If active order exists:
      - Display a warning message: "Vui lòng thanh toán đơn hàng trước khi đóng bàn."
      - Prevent closing action.
    - If no active order: Allow the close action.
  - Business Rules:
    - BR-01: Cannot close a table with unpaid or active orders.
    - BR-02: Table status must be updated in real-time to prevent double-booking or incorrect availability.

- Functionalities:
  - When "Close Table" is clicked:
    - Validate if there are active orders.
    - If no active orders: Sends axios.PUT request to update table status to "Closed". Updates the UI to reflect the table as "Closed" (e.g., card turns red, shows "Bàn đóng").
    - If active orders exist: Displays a modal or popup warning, blocking the close action until the order is settled.
  - After closing successfully: Refresh or update the table card to reflect the new status.
  - Error handling: Displays an error notification if the API call fails.

3.2.1.4 Lock Table

- **Function trigger:** This function is triggered when Manager logs in to their account, navigates to the Table Management section, and clicks the "**Khóa bàn**" button on a table that is currently in the "Open" state. Upon clicking the **"Khóa bàn"** button, a popup appears asking for a maintenance reason. After filling the reason and confirming, the table will be locked for maintenance.

- **Function description:**
  - Roles: Manager
  - Purpose: To lock a table for maintenance so that it cannot be used by customers until it manually unlocked
  - Interface:
    - Display Current Status: "Bàn mở" (Table Open)
    - Button: Maintenance ("Bảo trì") — orange/yellow button
    - Popup Form:
      - Title: “Khóa bàn [Table Code] để bảo trì”
      - Text area: enter maintenance reason(Required)
      - Buttons:
        - Cancel(“Hủy)
        - Confirm Maintenance Lock (“Xác nhận khóa bàn”)
  - Data Processing:
    - Input Validation: Maintenance reason must be provided.
    - API Call: Sends the table ID and maintenance reason to the backend to lock the table.
    - Feedback: Displays success or error messages based on the API response. Updates the UI to show the table is under maintenance.
- **Screen layout:**

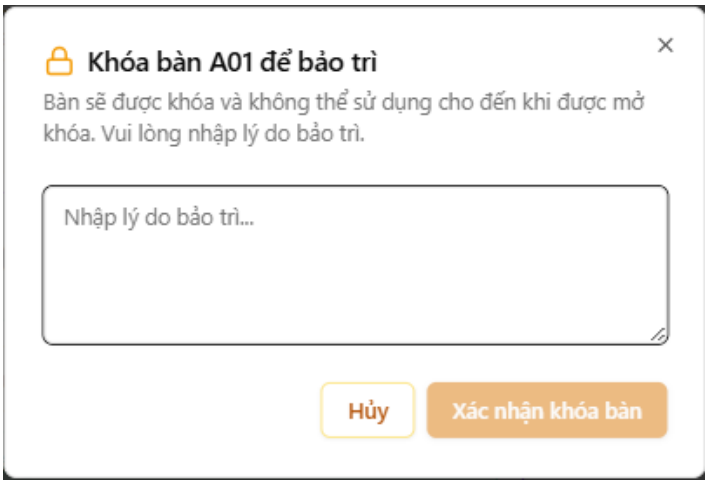


Figure 14- Lock Table

- **Function Details:**
  - Data:
    - TableId (string, required): The identifier of the table to be set under maintenance.
    - MaintenanceReason (string, required): The reason provided by the Manager for locking the table.
  - Validation:
    - Maintenance reasons are required and must not be empty.
    - Trims whitespace to ensure clean input.
  - Business Rules:
    - **BR-01**: A table under maintenance cannot be opened or assigned to customers.
    - **BR-02**: The maintenance reason must be stored along with the maintenance record for auditing purposes
  - Functionalities:
    - When the "Bảo trì" button is clicked: Open a popup asking for a maintenance reason.
    - When "Xác nhận khóa bàn" is clicked:
      - Validates the input.
      - If valid: Sends `axios.PUT` request to lock the table with the provided reason. Updates the UI: The table status changes to "Under Maintenance" or equivalent.
      - If invalid: Displays an error message asking the Managerto input the reason
      - The table cannot be booked, or used until it is manually unlocked.

3.2.1.5 Merge Table

- **Function trigger:** This function is triggered when Manager logs in, navigates to the “Bàn ăn“ section, and clicks on the **“Ghép bàn”** button.
  A popup panel appears where the Manager must:
  - Enter a **“Tên nhóm ghép bàn”**for the merged table group.
  - Select **only tables in "Closed" state** from the list.
    After selecting and clicking ”**Xác nhận ghép bàn**", the system will merge the selected tables into one group.
- **Function description:**
  - Role: Manager
  - Purpose: To group multiple closed tables into merged groups for serving larger parties or specific requirements.
  - Interface:
    - **Group Name** (Required): Text input to name the merged group.
    - **Search Field**: To filter/select tables by code.
    - **List of Tables**: Each item shows: table code, status : “Đang đóng”, capacity, checkbox to select for merger
    - Button: Cancel (“Hủy”) and Confirm Merger (“Xác nhận ghép bàn”) - only enable when group name is entered and at least 2 tables selected
  - Data Processing:
    - Input Validation: Group name is required and must not be empty. At least 2 closed tables must be selected
    - API Call: Calls API to fetch all tables with status “Closed” and display them. Send selected table IDs and group name to backend to process merge
    - Feedback: Success message if merge succeeds, UI updates to reflect group. Error message if merge fails due to invalid input or server issue
- **Screen layout:**

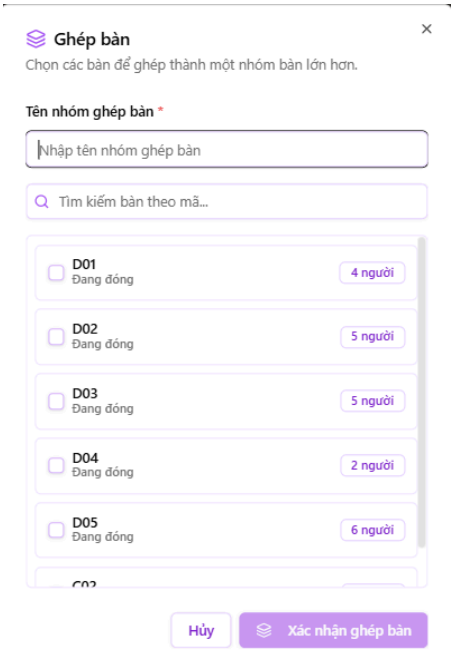


Figure 15- Merge Table

- **Function Details**
  - Data:
    - GroupName(string, required): The name for the new merge table group.
    - TableIds (array of string, required) : List of IDs for the selected closed tables to be merged
  - Validation:
    - Group name: Required and must not be empty. Trim whitespace before submission.
    - Table selection: Must select at least 2 tables. Only tables in status “Closed” can be listed/selected. UI must prevent selecting tables in any other status.
  - Business Rules
    - **BR-01**: Only closed tables can be merged.
    - **BR-02**: A table can only belong to one group at a time.
    - **BR-03**: Group name must be unique within the system.
  - Functionalities
    - Fetches all closed tables via API when modal is opened.
    - Filters tables in UI by code using search bar.
    - Disables "Confirm Merge" until group name is filled and valid table selection is made.
    - On submission:
      - Sends axios.PUT to backend with group name and selected tables
      - Handles API response:
        - On success: Merged group created, UI reflects new group
        - On failure: Displays proper error message
    - Resets form and closes popup upon success

3.2.1.6 Unmerge Table from Group

- **Function trigger:** This function is triggered when Manager logs in, navigates to the Table Management screen, views a merged table group, and clicks the "Hủy ghép bàn" button next to a specific table (e.g., A01). A popup appears asking for confirmation, and upon confirming, the selected table will be removed from the merged group.
- **Function description:**
  - Roles: Manager
  - Purpose: To remove a table from an existing merged group, restoring it to standalone usage.
  - Interface
    - **Trigger Location**: Inside merged group card view (e.g., A01 part of "Bàn ghép")
    - Popup Modal Fields:
      - Title: “Hủy ghép bàn [Table Code]”
      - Group Name: Display current group
      - Table Code: Table to be unmerged
      - Warning Text: "Hủy ghép bàn sẽ tách bàn này ra khỏi nhóm ghép bàn hiện tại."

        - Buttons: **Cancel** (“Hủy”) and **Confirm Unmerge** (“Xác nhận hủy ghép bàn”)

    - Data Processing
        - Input Validation: Table must belong to a merged group.
        - API Call: Sends request to backend to remove selected table from group.
        - Feedback:
            - Displays success message and updates table state in UI.
            - If failed, show descriptive error (e.g., "Không thể hủy ghép do đơn hàng đang phục vụ").

- **Screen layout:**

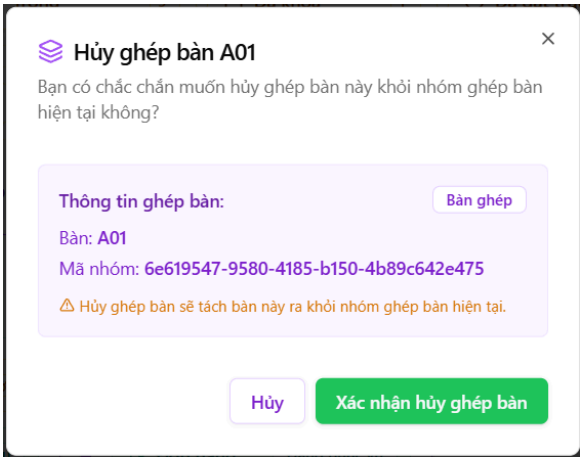


Figure 16- Cancel Merge Table

- **Function Details:**
    - Data:
        - TableId (string, required): The ID of the table to be removed.
        - GroupId (string, required): The ID of the merged group the table belongs to.
    - Validation:
        - Table must be part of a valid group.
        - Optional (Business enforced):
            - Prevent unmerge if the table is **in active service** (has order).
            - May show message like: “Không thể hủy ghép bàn khi đang phục vụ”.
    - Business Rules:
        - **BR-01**: A table can only be removed from a group if not actively in use, or allowed explicitly.
        - **BR-02**: If the group only has one table left after unmerge, the group may be auto-disbanded.
        - **BR-03**: Unmerge must update real-time table state and detach group metadata.
    - Functionalities:
        - When "Hủy ghép bàn" is clicked: Displays modal with table and group info.
        - When "Xác nhận hủy ghép bàn" is clicked:
            - Validates state.
            - Sends axios.PUT request to remove table from group.
        - UI is updated accordingly:
            - Table is visually removed from the group.
            - Group counter (Số bàn, Tổng người) is updated

3.2.2 Product Management

3.2.2.1 Create Product

- **Function trigger:** This function is triggered when the Manager logs in, navigates to the Food Management panel, and clicks the "Add New Food" button.
- A popup form appears where the Manager can fill in all details of the new food item. After clicking the "Save" button, the food item is submitted to the backend.
- **Function description:**
    - Roles: Manager
    - Purpose: To allow Manager to add a new food item with full attributes including name, price, description, image, category, kitchen type, optional sizes and additional options.
    - Interface:
        - Food Name (Required)
        - Base Price (VND) (Required)
        - Description (Optional)
        - Image Upload (Optional)
        - Category (Required - dropdown)
        - Product Type (Required - dropdown)
        - Size & Price Options (Optional): Add multiple sizes (e.g., S, M, L) with custom prices.
        - Additional Product Options (Optional - multi-select dropdown)
    - Data processing:
        - Input Validation:
            - Ensures required fields (name, price, category, product type) are filled.
            - Prices must be valid positive numbers
        - API Call: Sends the complete product object (including image URL if uploaded) to backend via axios.POST
        - Feedback:
            - Displays success or error message based on API response.
            - Closes modal and resets form on success.
- **Screen layout:**

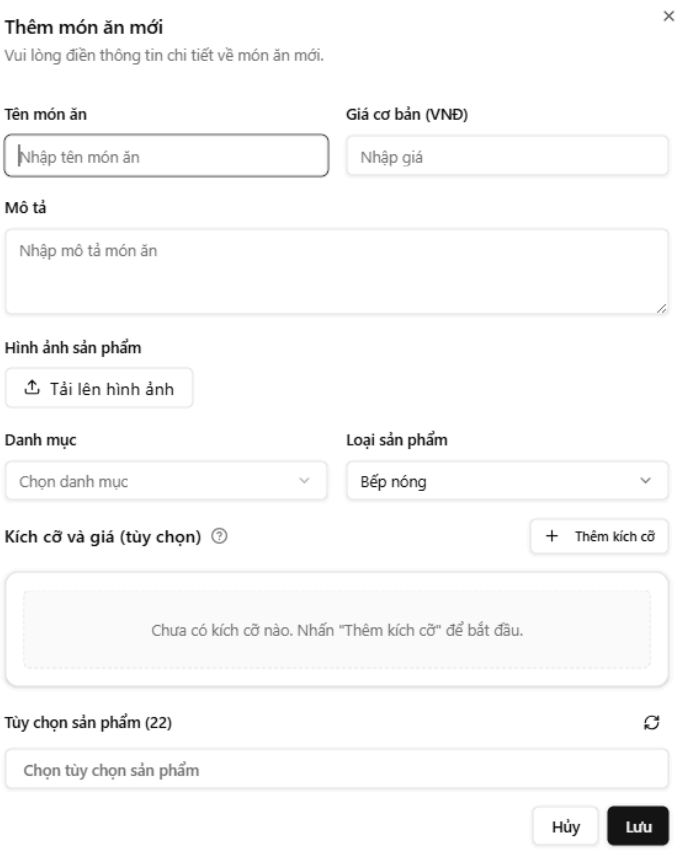


Figure 17- Manager Product Management

- **Function Details:**
  - Data:
    - Name (string, required): Name of the dish.
    - BasePrice (number, required): Default price in VND.
    - Description (string, optional): Textual description.
    - ImageUrl (string, optional): Image file uploaded and converted to URL.
    - CategoryId (string, required): Selected food category.
    - ProductType (enum, required): Kitchen type ("HotKitchen " or "ColdKitchen ").
    - Sizes (array, optional): Each item includes:
      - SizeName(string)
      - SizePrice(number)
    - AddOnOptions (array, optional): List of additional option IDs.
  - Validation:
    - Food Name: Required and must not be empty. Trims whitespace and limits length.
    - Price: Required, must be a valid number greater than 0.
    - Image: Optional, but must be valid format (jpg/png/webp)
    - Category and Product Type: Must be selected from predefined dropdown
    - Size List(Optional): Each size name must not be empty. Prices must be ≥ 0.
  - Business Rules:
    - **BR-01**: Food names must be unique within the same category.
    - **BR-02**: Each product must belong to one kitchen (hot/cold).

        - **BR-03**: Sizes and customizations are optional but must be valid if provided.
        - **BR-04**: Image, if uploaded, must be stored securely and referenced by URL.
        - a
    - Functionalities:
        - UI Form renders with all required input fields and dropdowns.
        - Manager fills in required details and clicks “Lưu”:
            - Validates all required fields and optional blocks
            - Sends all payload to backend via axios.POST
        - Handle message: Closes modal, show success message, and refreshes product list. If error display message (“ Lỗi tạo món ăn.” ,..)

3.2.2.2 Create Product Combo

- **Function trigger:** This function is triggered when the Manager logs in, navigates to the “ Thực đơn” or Product Management section, and clicks the "Tạo Combo" button. A popup form appears where the Manager can configure the combo name, price, image, category, and assign product groups. After clicking the "Tạo Combo" button, the combo is submitted to the backend..
- **Function description:**
    - Roles: Manager
    - Purpose: To allow managers to create a new food combo by grouping existing food items into structured product groups with optional extra pricing.
    - Interface:
        - Combo Name (Required)
        - Combo Price (VND) (Required)
        - Image Upload (Optional)
        - Description (Optional)
        - Category (Required — dropdown)
        - Product Groups within Combo:
            - Group Name (e.g., Pizza, Đồ uống, Món phụ) (Required)
            - Add food products to each group (multi-row)
            - Set additional price per product (if any)
        - Buttons:
            - Add Group(“Thêm nhóm”)
            - Add Product to Group (“Thêm sản phẩm”)
            - Save Combo (“Tạo combo”)
            - Cancel (“Hủy”)

    - Data Processing:
        - Input Validation:
            - Combo name, price, category, and at least one product group are required.
            - Products in each group must be selected and valid.
        - API Call: Sends a full combo object to the backend via axios.POST, including all product groups and sub-products.
        - Feedback:
            - On success: Closes modal, shows success message, refreshes combo list.

- On failure: Shows validation or server-side error message.

- **Screen layout:**

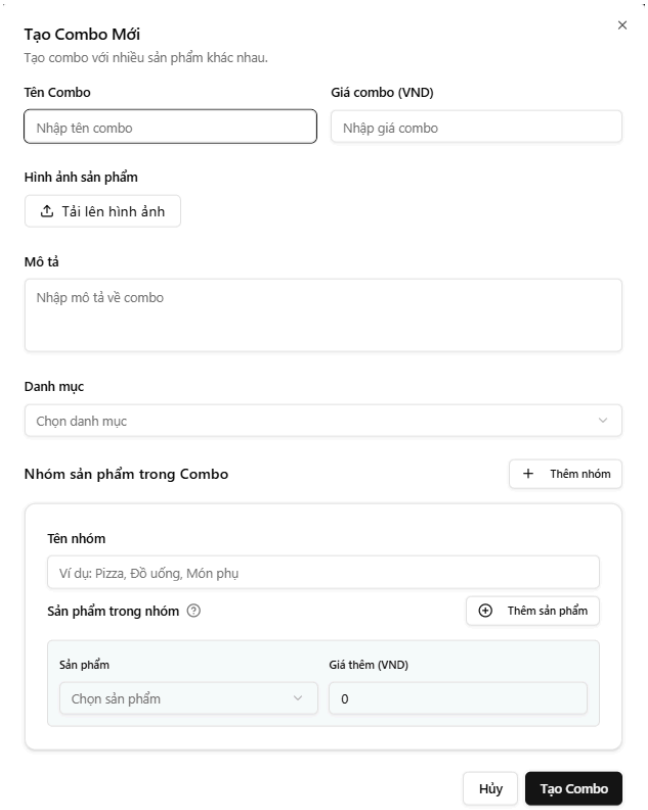


Figure 18- Create Product Combo

- **Function Details:**
  - Data:
    - ComboName (string, required): Name of the combo.
    - ComboPrice (number, required): Base price of the combo.
    - ImageUrl (string, optional): Uploaded product image.
    - Description (string, optional): Description of the combo.
    - CategoryId (string, required): Selected category from dropdown.
    - Groups (array, required):
      - Each group includes:
        - GroupName(string, required)
        - ProductsInGroup: array of: ProductId(string,required), ExtraPrice(number, optional)
  - Validation:
    - Combo Name: Required, trimmed, unique within category.
    - Combo Price: Must be a valid number > 0.
    - Image: Optional; validate format and file size.
    - Groups:
      - At least one group must exist
      - Each group must have a valid name and at least one product selected.

            - Extra price must be a number ≥ 0.
        - Prevent submission if any required fields are missing or invalid.
    - Business Rules:
        - BR-01: A combo must consist of at least one valid product group.
        - BR-02: Product used in combo must exist and be active.
        - BR-03: Group names must be unique within the combo.
        - BR-04: Combo price must reflect base value; extra prices are optional and dynamic.
    - Functionalities:
        - Manager inputs combo details and configures product groups.
        - When "Tạo Combo" is clicked:
            - Validate all input fields.
            - Compose full combo payload including nested group-product mappings.
            - Send API calls to create a combo.
        - Upon success:
            - Display success message.
            - Refresh or update combo product list in UI
        - On error: Display error with appropriate message

3.2.2.3 Update Status Product

- **Function trigger:** This function is triggered when the Manager opens the Product Detail screen by clicking on a specific product in the product list.
- On this detail screen, the Manager can select a status from a dropdown list (Có sẵn, Hết nguyên liệu, Đã khóa) and click "Cập nhật" to apply the new status..
- **Function description:**
    - Roles: Manager
    - Purpose: To update the availability status of a product (e.g., currently sellable, temporarily unavailable, or locked from the menu).
    - Interface:
        - “Có sẵn” (Available)
        - “Hết nguyên liệu” (Out of Stock)
        - “Đã khóa” (Locked)
        - Button: Triggers the update API Call
        - Visual indicators:
            - Status label changes color/icon based on selection
            - Displayed next to product category, price, and type
    - Data Processing:
        - Input Validation: Ensures a valid status is selected (one of the predefined options).
        - API Call: Sends a request to update the product’s Status via axios.PUT
        - Feedback:
            - Shows success message and refreshes UI if update is successful
            - Shows error message if update fails.
- **Screen layout:**

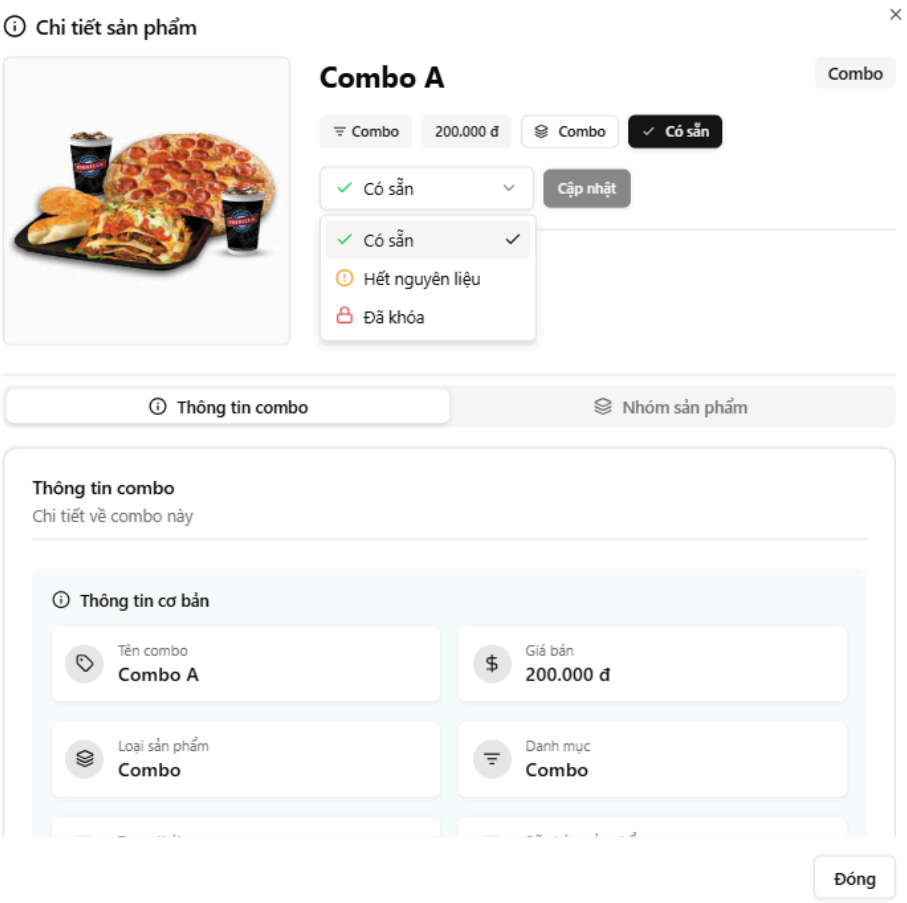


Figure 19- Update Status Product

- **Function Details:**
  - Data:
    - ProductId(string, required): Unique identifier of the product.
    - Status(enum, required): One of the following (Available, OutOfStock, Locked)
  - Validation:
    - Selected status must match one of the valid enum values.
    - Disable button or prevent submission if no change has been made.
  - Business Rules:
    - BR-01: Only Managers can change product status.
    - BR-02: Product in Locked status will be hidden from ordering UI
    - BR-03: Status change must be immediately reflected in menu listings and inventory logic.
  - Functionalities:
    - Upon loading product detail: Shows current product info including current status.
    - When manager changes dropdown value and clicks “Cập nhật”
      - Validates input.
      - Sends API requests to update product status.
      - Display toast or notification for success/failure.

3.2.3 Ingredient Management

3.2.3.1 Create Ingredient

- **Function trigger:** This function is triggered when the Manager accesses the “Thực Đơn” section from the sidebar and selects the “Nguyên Liệu” tab. In this view, clicking the “Thêm Nguyên Liệu” button opens a popup form where the Manager can input ingredient name and description. After clicking “Lưu”, the new ingredient is added to the system.
- **Function description:**
  - Roles: Manager
  - Purpose: To allow Manageristrators to add new ingredients that can later be assigned to food products or recipes.
  - Interface:
    - Tên Nguyên Liệu (Required): Text input for ingredient name.
    - Mô Tả (Required): Text area for a description of the ingredient.
    - Buttons:
      - Lưu: Triggers form submission
      - Hủy: Close the popup without saving
  - Data Processing:
    - Input Validation: Ingredient name and description must both be provided. Trims whitespace and checks for duplicates.
    - API Call: Sends a POST request to backend
    - Feedback:
      - Displays a success notification if the ingredient is added.
      - Displays error message for duplicate or invalid data.
- **Screen layout:**

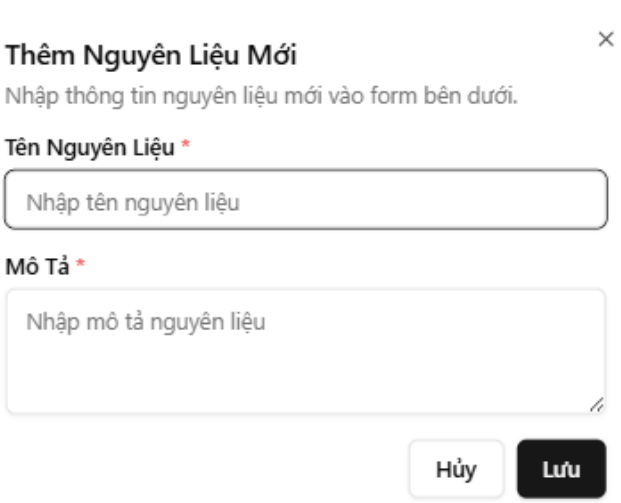


Figure 20-Create Ingredient

- **Function Details:**
  - Data:
    - IngredientName (string, required): Name of the ingredient to be created.
    - Description (string, required): Description or note about the ingredient (e.g., taste, usage, storage).
  - Validation:

        - Both fields are required.
        - Ingredient names must be unique in the system.
    - Business Rules:
        - **BR-01**: Ingredient names must be unique (case-insensitive).
        - **BR-02**: Ingredient records must be used later in recipes or inventory mapping.
    - Functionalities:
        - Clicking **"Thêm Nguyên Liệu"** opens a centered modals.
        - Manager enters required fields and clicks **"Lưu":**
            - Performs client-side validation.
            - Sends data via API to the backend.
            - On success, closes modal, adds new entry to table of ingredients. On failure, display validation or API error.

3.2.4 Category Management

3.2.4.1 Create Category

- **Function trigger**: This function is triggered when the Manager opens the "Loại" tab inside the "Quản lý thực đơn" section and clicks on the "Thêm danh mục" button.
- A popup form appears, allowing the Manager to input a new category name and description. After clicking "Lưu", the new category is added to the system.
- **Function description:**
    - Roles: Manager
    - Purpose: To allow Manager to create new food categories for organizing dishes in the menu (e.g., Pizza, Món tráng miệng, Món khai vị, etc.)
    - Interface:
        - Tên Danh Mục (Required): Text input for the name of the new category.
        - Mô Tả (Optional): Text area to describe the category.
        - Buttons:
            - Lưu: Submits the form.
            - Hủy: Closes the popup without saving.
    - **Data Processing:**
        - Input Validation:
            - Name is required, must not be empty or duplicate.
            - Description is optional but must be trimmed and within reasonable length.
        - API Call : Sends a POST request to backend
        - Feedback:
            - On success: Adds category to the list, closes popup, shows success message.
            - On failure: Displays validation or API error message (e.g., duplicate name).
- **Screen layout:**

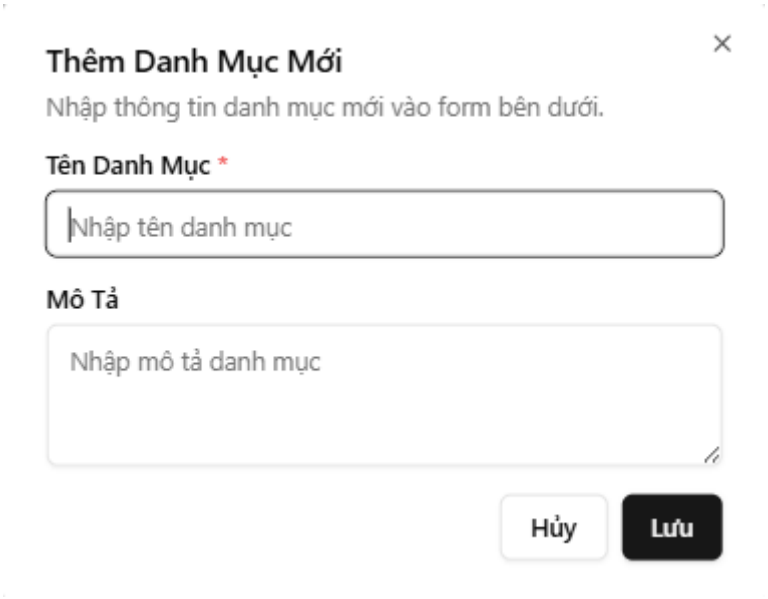


Figure 21 - Create Category

- **Function Details:**
  - Data:
    - CategoryName (string, required): Name of the new category (e.g., Pizza, Combo).
    - Description (string, optional): Description or note to explain the type of dishes in this category.
  - Validation:
    - Tên Danh Mục: Required, trim whitespace, must be unique (case-insensitive).
    - Mô tả: Optional, but should be limited in length (e.g., max 255 characters).
  - Business Rules:
    - BR-01: Category name must be unique within the system.
    - BR-02: Categories are used to organize food items in creation/edit screens.
  - Functionalities:
    - Clicking “Thêm danh mục” opens the modal form
    - Manager fills in required fields and clicks “Lưu”
      - Performs client-side validation.
      - Sends data to backend.
      - On success: Updates category list without page reload.
      - On error: Displays error message, keeps modal open for correction.

3.2.4.2 Update Category

- **Function trigger:** This function is triggered when the Manager accesses the **"Loại"** tab in **Thực Đơn** section and clicks the **edit (** ✎ **)** icon next to a category.
- **Function description:**
  - Roles: Manager
  - Purpose: To allow authorized users to modify an existing food category, including name and description.
  - Interface:
    - Tên Danh Mục (Required): Pre-filled with the current category name.

        - Mô Tả (Optional): Pre-filled description of the category.
        - Buttons:
            - Lưu: Submit the updated information
            - Hủy: Cancel the updated and close the modal
    - Data Processing:
        - Input Validation: Name field is required and cannot be empty.
        - API Call: Sends a PUT request to the backend
        - Feedback: Shows success message and updates UI upon success. Displays error message if validation fails or server returns error.

- Screen layout:

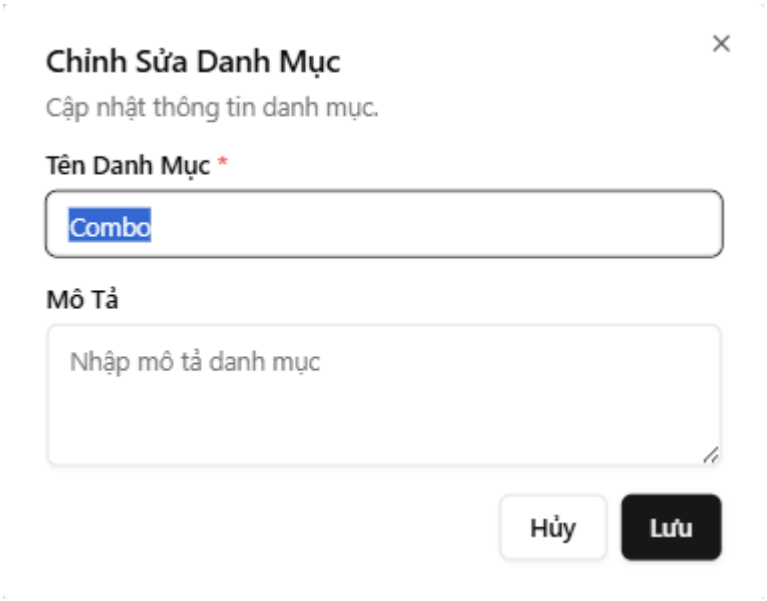


Figure 22- Update Status Category

- **Function Details:**
    - Data:
        - CategoryName (string, required): The updated name of the category.
        - Description (string, optional): The updated description.
    - Validation:
        - Tên danh mục: Required, trimmed input
        - Mô tả: Optional
    - Business Rules:
        - BR-01: Category name must remain unique across the system.
        - BR-02: Changes should immediately reflect in all related product forms.
        - BR-03: Categories used by existing products cannot be deleted, but can be updated.
    - Functionalities:
        - Open form with current category data.
        - Allows inline editing of fields.

- On clicking "Lưu": Validates fields, sends update to backend. On success: Closes modal, updates table, On error: Displays feedback, keeps modal open.

3.2.4.3 Remove Category

- **Function trigger:** This function is triggered when a Manager navigates to the **"Loại"** tab inside the **Thực đơn** section and clicks the **delete icon (**  **)** next to a specific category.
- A confirmation popup appears. If the user clicks "**Xóa**", the system proceeds to delete the selected category
- **Function description:**
  - Roles: Manager
  - Purpose: To permanently remove an existing food category from the system.
  - Interface Popup Title, Popup Content, Buttons
  - Data Processing:
    - Input validation: Ensure that category exists,
    - API Call: Sends a DELETE request to backend
    - Feedback: On success: A success message is shown, the popup closes, and the UI refreshes. On failure: An error message is shown (e.g., if the category is in use).
- **Screen layout:**

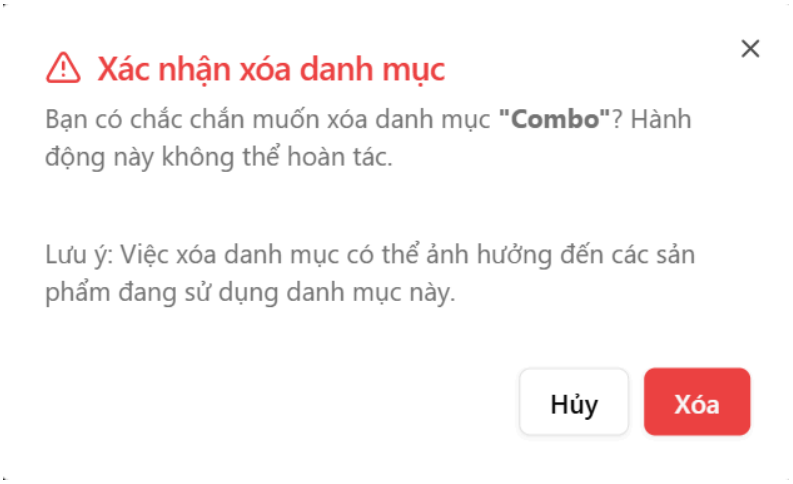


Figure 23- Update Status Category

- **Function Details:**
  - Data: CategoryId (string, required): The unique ID of the category to be deleted.
  - Validation:
    - Must ensure the category is not being actively referenced by products (unless soft delete is supported).
    - The system must warn the user that the deletion is irreversible and may affect existing items.
  - Business Rules:
    - BR-01: Cannot delete a category if it is linked to existing products, unless soft delete or cascading delete is implemented.

        - BR-02: Deletion is permanent and irreversible.
        - BR-03: Once deleted, the category name may be reused if a new category is created.
    - Functionalities:
        - Clicking the delete icon opens the popup.
        - The system shows a warning message including the category name.
        - If the user clicks "Xóa":
            - Send a request to the backend.
            - On success: The modal closes, the category list is updated.
            - On error: An error message appears in the popup or notification.
        - Clicking "Hủy" closes the popup without deleting the category.

3.2.5 Order Management

3.2.5.1 Place order

- **Function trigger:** This function is triggered when the Manager logs into the system, navigates to the order screen for a table, and clicks the "Đặt đơn" (Place Order) button.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** To confirm and place the order for the selected table based on items currently in the cart.
    - **Interface:**
        - Displays current selected items (with image, name, quantity, and total price).
        - Displays the "Thêm món" (Add Item) and "Đặt đơn" (Place Order) buttons at the bottom.
    - **Data Processing:**
        - **Input Validation:** Validates that a valid tableId exists and has an active Order.Validates that order exists and matches the expected unpaid status. Validates that products and option items associated with the order exist.
        - **API Call:** Sends the finalized order details to the backend to update the order status from cart to confirmed order.
        - **Feedback:** Displays success or error messages based on the API response.
- **Screen layout:**

Figure 24- Manager Order Management

- **Function Details:**
  - Data:
    - **tableId** (string, required): The identifier for the table (e.g., B02).

        - **orderId** (string, required): The current active order linked to the table.
        - **productId** (string, required): The identifier for each ordered product.
        - **optionItemIds** (array of strings, optional): List of selected product option IDs (e.g., toppings, sauces).
    - Validation:
        - **tableId:**
            - Must exist in the system.
            - Must have a CurrentOrderId assigned.
            - Error if table not found → "Table does not exist".
        - **orderId:**
            - Must exist and be valid.
            - Order must be in "Unpaid" status.
        - **optionItemIds:**
            - If selected, each must exist.
    - Business Rule:
        - **BR-27**: Order code must be generated using the last 9 digits of ticks (timestamp) + 3 random digits
        - **BR-49**: Orders cannot be placed for a table that has been disabled by the Manager unless the table is occupied.
    - Functionalities:
        - Validate all inputs before sending the order.
        - Block order placement if validation fails, with clear error messages.
        - Use Axios.POST to send the order placement request to the backend.
        - Display loading animation while waiting for server response.
        - After successful placement, refresh the screen to show the new order under "ĐƠN ĐÃ ĐẶT" tab.
        - Reset the cart view upon successful placement.

3.2.5.2 Check out order

- **Function trigger:** This function is triggered when the Manager views an active unpaid order and clicks the “Thanh toán” (Checkout/Payment) button.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Finalizes an order and marks it ready for payment. All ordered items must be either completed (done) or canceled. Any discounts and VAT are applied before final total calculation.
    - **Interface:**
        - Displays table details (capacity, area, last update).
        - Display field to insert voucher and a button to apply
        - Shows list of ordered items with quantity, price, and status.
        - Shows status tags: “Đã Check Out”if order has been processed.
        - Shows applied VAT and calculated total.
        - Displays the following action buttons:

- **Hủy đơn hàng** (Cancel Order)
- **Thanh toán** (Checkout)
- **Hủy Checkout** (Revert checkout)

    - **Data Processing:**
        - **Input Validation:** Validates that order exists and matches the expected unpaid status. Validates that products and option items associated with the order exist. All selected option items must be valid. VAT is added to the subtotal. Voucher exists if applied.
        - **API Call:** On clicking "Thanh toán", system sends a checkout request to backend, passing:
            - orderId
            - list of finalized OrderItems
            - applied voucherId (if any)
            - VAT config
            - voucherId
        - **Feedback:** Displays success or error messages based on the API response.
- **Screen layout:**

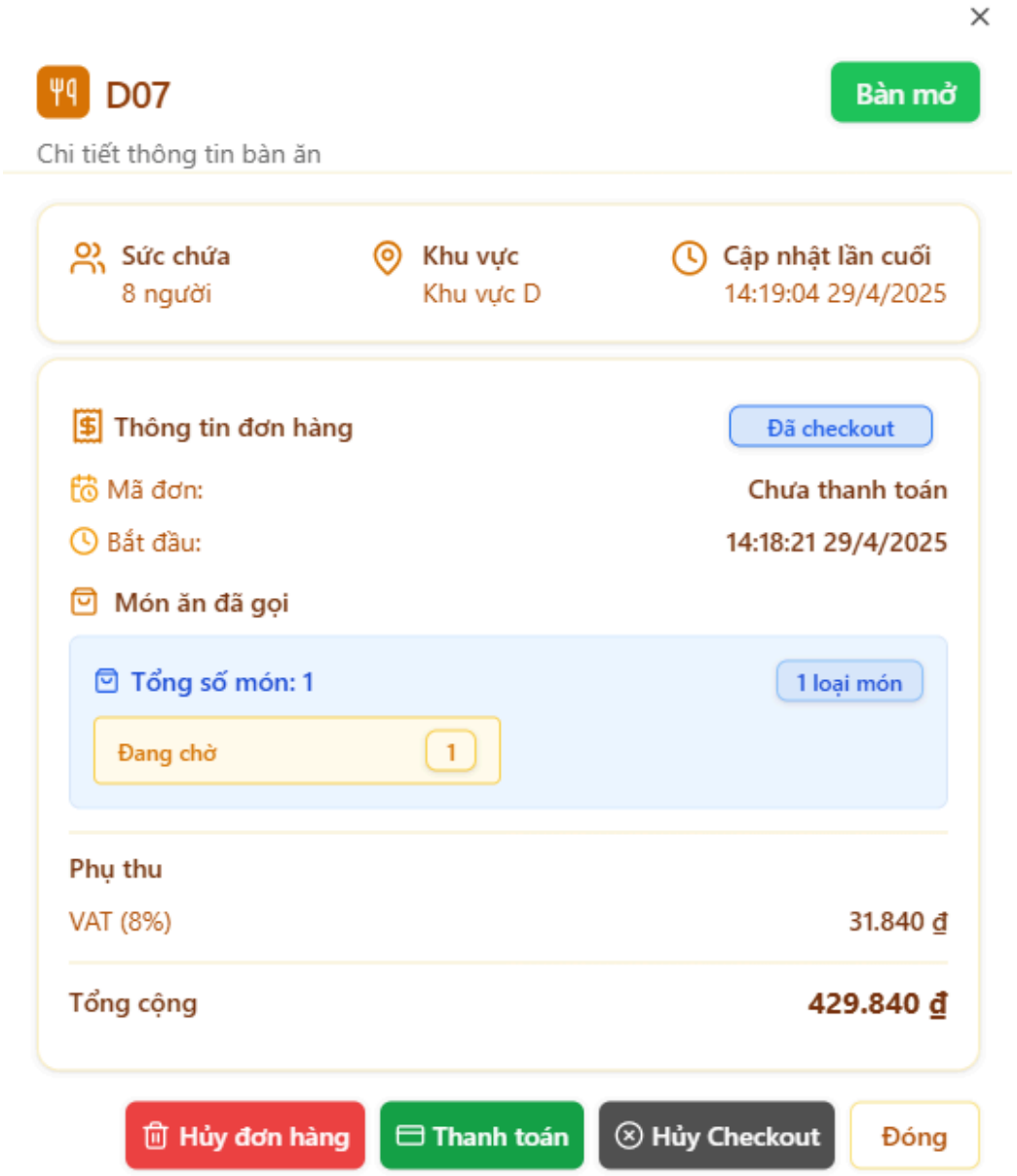

D07
Bàn mở
Chi tiết thông tin bàn ăn
Sức chứa
8 người
Khu vực
Khu vực D
Cập nhật lần cuối
14:19:04 29/4/2025
Thông tin đơn hàng
Đã checkout
Mã đơn:
Chưa thanh toán
Bắt đầu:
14:18:21 29/4/2025
Món ăn đã gọi
Tổng số món: 1
1 loại món
Đang chờ
1
Phụ thu
VAT (8%)
31.840 ₫
Tổng cộng
429.840 ₫
Hủy đơn hàng
Thanh toán
Hủy Checkout
Đóng

Figure 25.1- Check out order in Staff site

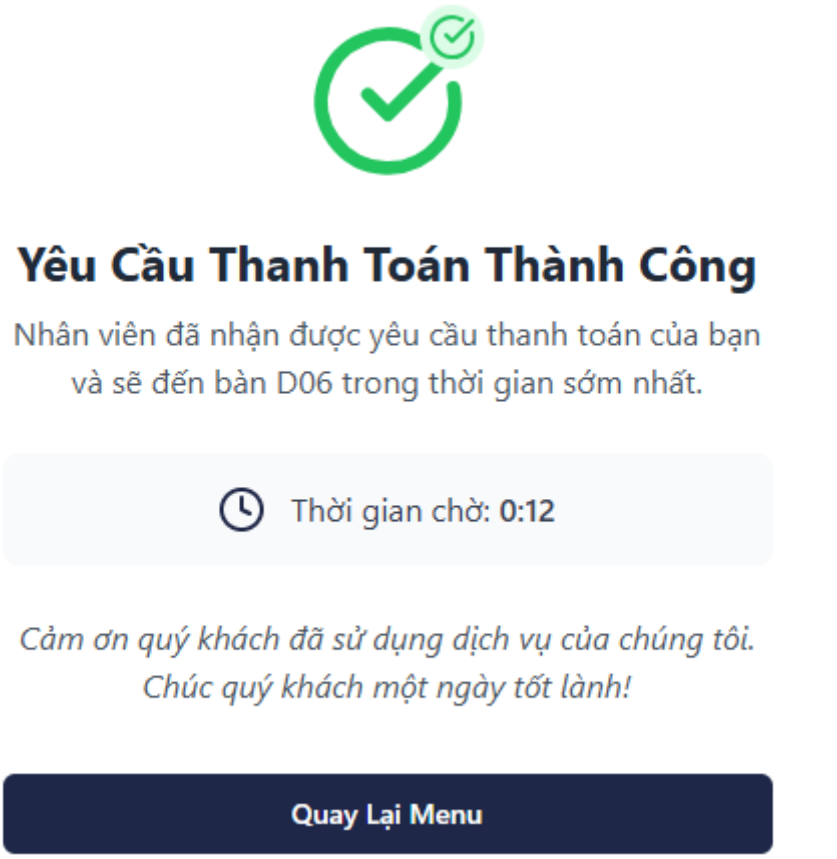


Figure 25.2- Check out order in Customer site

- **Function Details:**
  - Data:
    - **orderId**(string, required): The identifier for the table (e.g., B02).
    - **OrderItems**(string, required): The current active order linked to the table.
    - **voucherId**(string, optional): The identifier for each ordered product.
    - **VAT**(decimal): List of selected product option IDs (e.g., toppings, sauces).
  - Validation:
    - **orderId:** Must exist and be in unpaid status
    - **OrderItems:** Must be all “Done” or “Cancelled”
    - **VoucherId:** If present, must match known voucher with valid type
    - **AdditionalFee:** Configed for VAT
  - Business Rule:
    - **BR-27**: Order code must using the last 9 digits of ticks (timestamp) + 3 random digits
    - **BR-28–30:** Voucher existence, application rules (direct and %), and validity date check
    - **BR-49**: Orders cannot be placed for a table that has been disabled by the Manager unless the table is occupied.
    - **BR-50:** Voucher usage tracked for customer to prevent abuse
  - Functionalities:

        - Validate all inputs before sending the order.
        - Block order check out if validation fails, with clear error messages.
        - Use Axios.POST to send the order placement request to the backend.
        - Display loading animation while waiting for server response.
        - Reset the order view upon successful placement.

3.2.5.3 Cancel check out order

- **Function trigger:** This function is triggered when the Manager views an active unpaid order and clicks the “Thanh toán” (Checkout/Payment) button.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Finalizes an order and marks it ready for payment. All ordered items must be either completed (done) or canceled. Any discounts and VAT are applied before final total calculation.
    - **Interface:**
        - Displays table details (capacity, area, last update).
        - Display field to insert voucher and a button to apply
        - Shows list of ordered items with quantity, price, and status.
        - Shows status tags: “Đã Check Out”if order has been processed.
        - Shows applied VAT and calculated total.
        - Displays the following action buttons:
            - **Hủy đơn hàng** (Cancel Order)
            - **Thanh toán** (Checkout)
            - **Hủy Checkout** (Revert checkout)
    - **Data Processing:**
        - **Input Validation:** Validates that order exists and matches the expected unpaid status. Validates that products and option items associated with the order exist. All selected option items must be valid. VAT is added to the subtotal. Voucher exists if applied.
        - **API Call:** On clicking "Thanh toán", system sends a checkout request to backend, passing:
            - orderId
            - list of finalized OrderItems
            - applied voucherId (if any)
            - VAT config
            - voucherId
        - **Feedback:** Displays success or error messages based on the API response.
- **Screen layout:**

Figure 26.1- Cancel check out order in Customer site

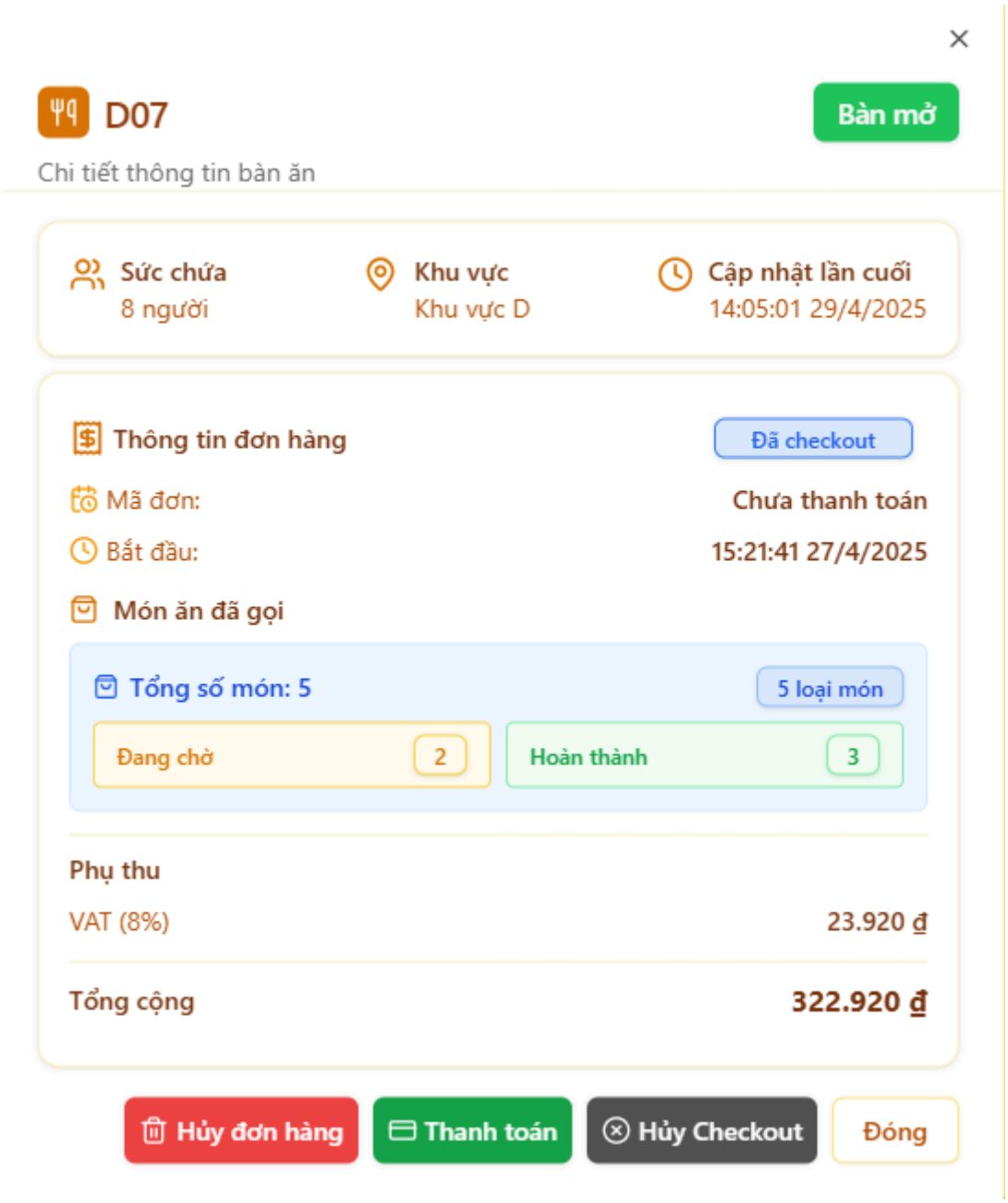


Figure 26.2- Cancel check out order in Customer site

- **Function Details:**
  - Data:
    - **tableId** (string, required): The identifier for the table (e.g., B02).
    - **orderId** (string, required): The current active order linked to the table.
    - **productId** (string, required): The identifier for each ordered product.
    - **optionItemIds** (array of strings, optional): List of selected product option IDs (e.g., toppings, sauces).
  - Validation:
    - **tableId:**
      - Must exist in the system.
      - Must have a CurrentOrderId assigned.
      - Error if table not found → "Table does not exist".

        - **orderId:**
            - Must exist and be valid.
            - Order must be in "Unpaid" status.
        - **optionItemIds:**
            - If selected, each must exist.
    - Business Rule:
        - **BR-27**: Order code must be generated using the last 9 digits of ticks (timestamp) + 3 random digits
        - **BR-49**: Orders cannot be placed for a table that has been disabled by the Manager unless the table is occupied.
    - Functionalities:
        - Validate all inputs before sending the order.
        - Block order placement if validation fails, with clear error messages.
        - Use Axios.POST to send the order placement request to the backend.
        - Display loading animation while waiting for server response.
        - After successful placement, refresh the screen to show the new order under "ĐƠN ĐÃ ĐẶT" tab.
        - Reset the cart view upon successful placement.

3.2.5.4 Swap table for order

- **Function trigger:** This function is triggered when the Manager selects a table with an active order and clicks the “Chuyển bàn” (Transfer Table) button. A new table selection modal opens, allowing the Manager to select a target table for the transfer.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows the transfer of an active order from one table to another eligible, unoccupied table. Ensures data consistency and operational integrity across table assignments.
    - **Interface:**
        - Modal window titled "Chuyển bàn BXX" (e.g., B02).
        - Search field to filter available tables by code.
        - List of available tables displayed with:
            - Table code (e.g., D01)
            - Status ("Đang đóng có thể chuyển" – Closed, transferable)
            - Capacity (e.g., 4 người)
        - Action buttons:
            - Hủy (Cancel)
            - Xác nhận chuyển bàn (Confirm Transfer)
    - **Data Processing:**
        - **Input Validation:** Validates that order exists and matches the expected unpaid status. Validates that table exists. Only active, unpaid orders can be valid. VAT is added to the subtotal. Voucher exists if applied.
        - **API Call:** On “Xác nhận chuyển bàn”, the system sends a request to backend with:
            - sourceTableId
            - targetTableId
            - orderId
            - Validates order and both table IDs
            - Ensures target table is eligible (not occupied)
            - Reassigns order from source to target

        - Logs this action (optional)
    - **Feedback:** Displays success or error messages based on the API response.
- **Screen layout:**

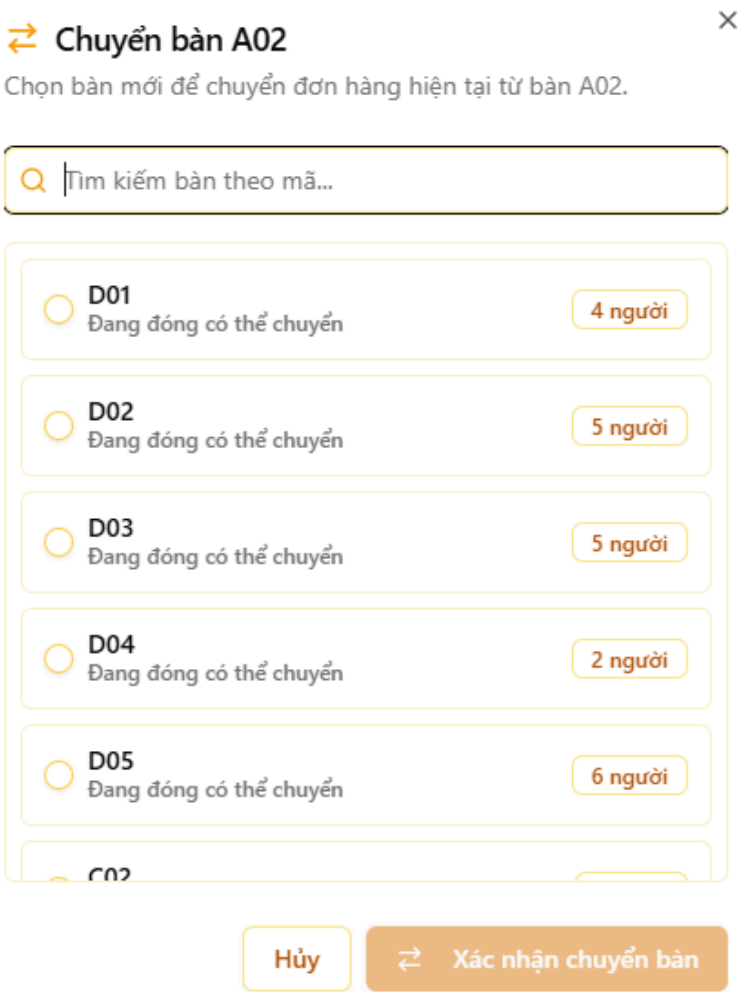


Figure 27- Swap table for order

- **Function Details:**
    - Data:
        - **sourceTableId**(string, required): Original table where order exists.
        - **targetTableId**(string, required): Selected table for transfer.
        - **orderId**(string, required): ID of the order to transfer.
    - Validation:
        - Order must be active and unpaid.
        - Target table must be valid and unoccupied.
        - Cannot transfer to the same table.
        - System must update all internal references to new table ID.
    - Business Rule:
        - **BR-49**: Managers and Staffs have the right to disable or enable table if needed, unless the table is occupied by customers
    - Functionalities:
        - Validate all inputs before sending the order.
        - Block order placement if validation fails, with clear error messages.
        - Use Axios.POST to send the order placement request to the backend.

        - Display loading animation while waiting for server response.
        - After successful swapping, refresh the screen to show the new table information.
        - Reset the table view upon successful placement.

3.2.5.5 Create payment QR Code

- **Function trigger:** This function is triggered when the Manager selects a table with an active order and clicks the “Chuyển bàn” (Transfer Table) button. A new table selection modal opens, allowing the Manager to select a target table for the transfer.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows the transfer of an active order from one table to another eligible, unoccupied table. Ensures data consistency and operational integrity across table assignments.
    - **Interface:**
        - Modal window titled "Chuyển bàn BXX" (e.g., B02).
        - Search field to filter available tables by code.
        - List of available tables displayed with:
            - Table code (e.g., D01)
            - Status ("Đang đóng có thể chuyển" – Closed, transferable)
            - Capacity (e.g., 4 người)
        - Action buttons:
            - Hủy (Cancel)
            - Xác nhận chuyển bàn (Confirm Transfer)
    - **Data Processing:**
        - **Input Validation:** Validates that order exists and matches the expected unpaid status. Validates that table exists. Only active, unpaid orders can be valid. VAT is added to the subtotal. Voucher exists if applied.
        - **API Call:** On “Xác nhận chuyển bàn”, the system sends a request to backend with:
            - sourceTableId
            - targetTableId
            - orderId
            - Validates order and both table IDs
            - Ensures target table is eligible (not occupied)
            - Reassigns order from source to target
            - Logs this action (optional)
        - **Feedback:** Displays success or error messages based on the API response.
- **Screen layout:**

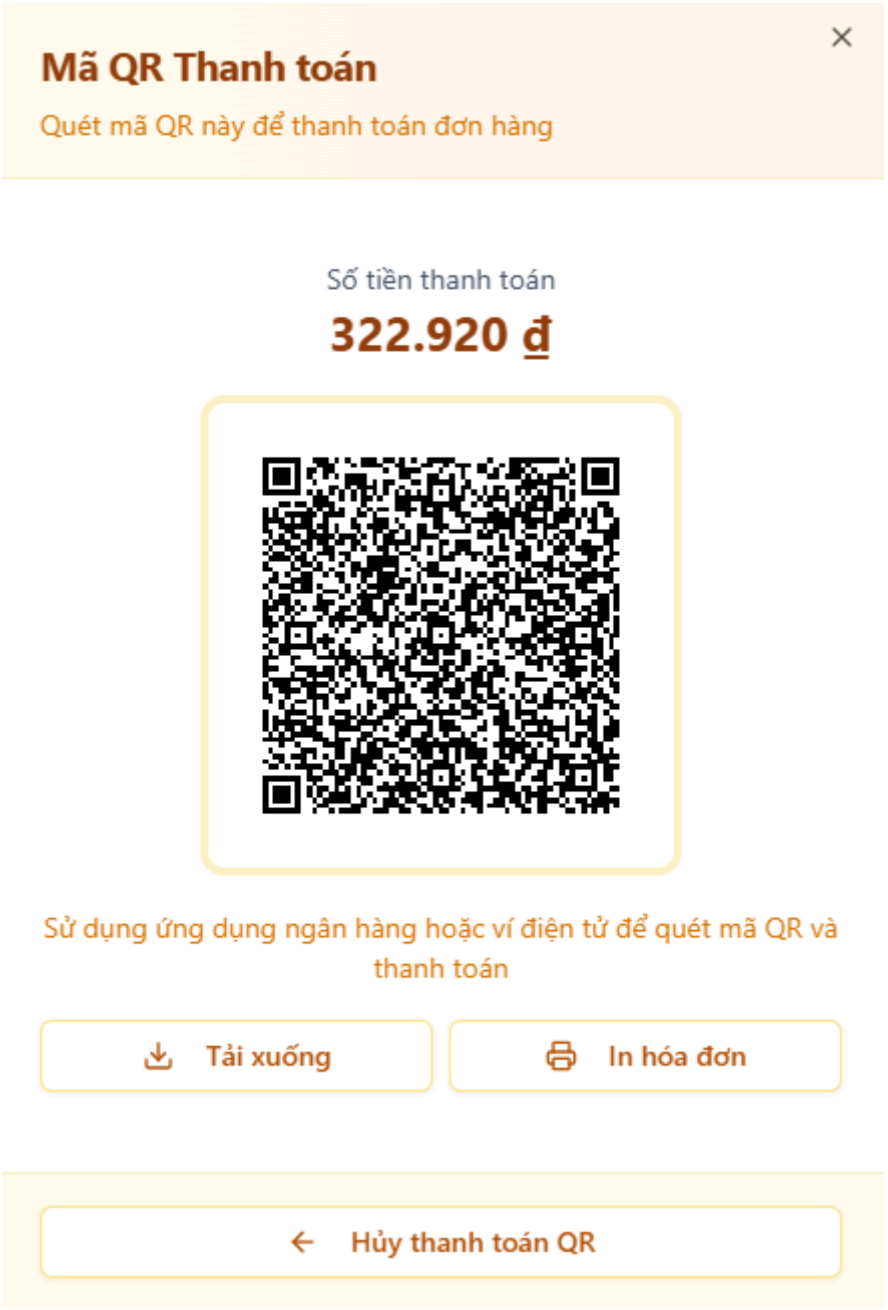


Figure 28- Create payment QR Code

- **Function Details:**
  - Data:
    - **sourceTableId**(string, required): Original table where order exists.
    - **targetTableId**(string, required): Selected table for transfer.
    - **orderId**(string, required): ID of the order to transfer.
  - Validation:
    - Order must be active and unpaid.
    - Target table must be valid and unoccupied.
    - Cannot transfer to the same table.
    - System must update all internal references to new table ID.
  - Business Rule:
    - **BR-49**: Managers and Staffs have the right to disable or enable table if needed, unless the table is occupied by customers

- Functionalities:
  - Validate all inputs before sending the order.
  - Block order placement if validation fails, with clear error messages.
  - Use Axios.POST to send the order placement request to the backend.
  - Display loading animation while waiting for server response.
  - After successful swapping, refresh the screen to show the new table information.
  - Reset the table view upon successful placement.

3.2.5.6 Cancel payment QR Code

- **Function trigger:** This function is triggered when the cashier clicks the "Hủy thanh toán QR" (Cancel QR Payment) button in the payment interface.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows authorized personnel to cancel a QR code payment that has not yet been completed. This action reverts the order state and removes the pending QR payment entry.
  - **Interface:**
    - QR code payment screen
    - "Hủy thanh toán QR" button
    - Confirmation message upon successful cancellation
  - **Data Processing:**
    - **Input Validation:**
      - Ensure the payment exists and is associated with the order
      - Confirm the payment method is QR and the payment has not been finalized
    - **API Call:** Sends a request to cancel the QR payment and revert any affected states
    - **Feedback:** Displays a confirmation message.
- **Screen layout:**

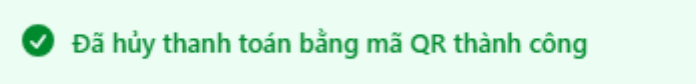


Figure 29- Cancel payment QR Code

- **Function Details:**
  - Data:
    - **paymentId (GUID, required):** Unique identifier of the QR payment record
    - **orderId (GUID, required):** Identifier of the related order
    - **PaymentMethod (enum):** QR
  - Validation:
    - QR payment must exist
    - QR payment must not be marked as completed
    - Order must be in a valid status for payment cancellation
  - Business Rule:
    - **BR-12:** Only authenticated users with valid roles can perform actions
    - **BR-51:** The system needs to record a history of all changes made by the Manage
  - Functionalities:
    - Reverts the order to the previous unpaid state

        - Cancels the pending QR payment record
        - Updates the UI to reflect that no payment is currently in progress
        - Logs the cancellation in the system for auditing
        - Notifies the user that the cancellation has been processed successfully

3.2.5.7 Pay order by Cash

- **Function trigger:** This function is triggered when the cashier selects “Thanh toán tiền mặt” (Pay by Cash) in the order payment interface.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows authorized personnel to complete an order payment using the cash method. This process also updates voucher usage, marks the order as paid, and resets the table’s state.
    - **Interface:**
        - Payment screen with two options:
        - Thanh toán tiền mặt (Pay by Cash)
        - Thanh toán chuyển khoản (Bank Transfer)
        - Confirmation display of total amount and successful payment message.
    - **Data Processing:**
        - **Input Validation:** Check if the order exists
        - Ensure the order is in the 'CheckedOut' status
        - Validate all applied vouchers are in usable condition
        - **API Call:** On “Xác nhận chuyển bàn”, the system sends a request to backend with:
            - sourceTableId
            - targetTableId
            - orderId
            - Validates order and both table IDs
            - Ensures target table is eligible (not occupied)
            - Reassigns order from source to target
            - Logs this action (optional)
        - **Feedback:** Displays success or error messages based on the API response.
- **Screen layout:**

Figure 30- Pay order by Cash

- **Function Details:**
  - Data:
    - **sourceTableId**(string, required): Original table where order exists.
    - **targetTableId**(string, required): Selected table for transfer.
    - **orderId**(string, required): ID of the order to transfer.
  - Validation:
    - Order must exist and be in the correct status (CheckedOut)
    - Voucher(s) used must be valid and available
    - Table must be linked to the order being paid
  - Business Rule:
    - **BR-12:** Only authenticated users with valid roles can perform actions
    - **BR-50:** The system will track voucher redemption history per customer to prevent fraudulent use
    - **BR-51:** The system needs to record a history of all changes made by the Manager
  - Functionalities:
    - Sets order status to 'Paid'
    - Updates voucher status to 'Used' if applicable
    - Releases the table(s) by setting CurrentOrderId to null and updating their status to 'Closing'
    - If table belongs to a merged group, all linked tables are updated

- Deletes merge group if it exists
- Adds a new payment record in the system
- Shows confirmation of successful payment to the user

3.2.6 Order item management

3.2.6.1 Create order item

- **Function trigger:** This function is triggered when the Manager clicks the “Đặt đơn” button on the cart page after selecting menu items.
- **Function description:**
  - **Roles:** Customer
  - **Purpose:** Allows the Customer to create a new order linked to a selected table. Ensures only one active order per table unless tables are merged.
  - **Interface:**
    - Button: "Đặt đơn" (Submit Order)
    - Confirmation Modal: Confirms intent to place the order
    - Displays current table, selected items, total price, and any modifiers or options selected
  - **Data processing**:
    - **Input Validation**:
      - Ensure that the table ID is valid
      - Ensure the table has not already been assigned a current order (unless it is part of a merged table group)
    - **API Call**: Sends the staff details to the backend for storage
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

Figure 31- Create order item

- **Function Details:**
  - Data:
    - **tableId (GUID, required):** Unique identifier of the table being ordered at
    - **type (enum, required):** Order type (e.g., dine-in, takeaway)
  - Validation:
    - **Ensure the table exists and is not already assigned an active order**
    - **If part of a merged group, update all related tables**
  - Business rules:
    - **BR-12:** Only authenticated users with valid roles can perform actions
    - **BR-36:** A reservation must have a table assigned before checking in
    - **BR-51:** The system needs to record a history of all changes made by the Manager
  - Functionalities:
    - Creates an order with the selected table and order type
    - If the table is merged (via TableMergeId), updates all related tables to use the new order ID
    - Stores the newly created order in the backend
    - Updates the table's state to reflect the ongoing order

- Displays confirmation and success message to the user
- Handles exceptions for duplicated order creation or invalid state

3.2.6.2 Confirm to cook order item

- **Function trigger:** Triggered when the user (typically a Chef) clicks the “Xác nhận nấu” (Start Cooking) button on a pending order item in the kitchen interface.
- **Function description:**
  - **Roles:** Chef
  - **Purpose:** Updates the order item’s status to Cooking, indicating that preparation has started.
  - **Interface:**
    - Displayed on active order cards
    - Button labeled “Xác nhận nấu” appears when item is in “Order” status
    - Status badge changes to “Đang nấu” once confirmed
  - **Data processing**:
    - **Input Validation**: Ensures that username, password, fullname, email, phone number, type and status are provided and valid.
    - **API Call**: Sends the staff details to the backend for storage
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

Figure 32- Confirm to cook order item

- **Function Details:**
  - Data:
    - **OrderItemId (GUID, required):** Unique ID of the item being cooked
    - **CookingStartTime (datetime, system-generated):** Timestamp recorded when item is confirmed for cooking
  - Validation:

        - **Cannot mark an item as done if already served or cancelled**
    - Business rules:
        - **BR-12:** Each user only has access rights and performs activities according to their assigned role
        - **BR-51:** The system needs to record a history of all changes made by the Manager to information products, configurations and other data
    - Functionalities:
        - Enables chefs to confirm preparation start for an item
        - Automatically records the cooking start timestamp
          Triggers a real-time status update visible to kitchen and service staff
          Prevents further edits or cancellation unless explicitly allowed by system rules

3.2.6.3 Done cooking order item

- **Function trigger:** Triggered when the user (typically a Chef) clicks the “Đã hoàn thành” (Done Cooking) button from the order item panel in the kitchen interface.
- **Function description:**
    - **Roles:** Chef
    - **Purpose:** Marks a food item as fully cooked and ready for serving, transitioning it from the kitchen phase to the serving queue.
    - **Interface:**
        - Displayed on active order cards in the kitchen interface.
        - Includes order item name, quantity, cook status, and cooking start time.
    - **Data processing**:
        - **Input Validation**: Ensures that username, password, fullname, email, phone number, type and status are provided and valid.
        - **API Call**: Sends the staff details to the backend for storage
        - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

Figure 33- Done cooking order item

- **Function Details:**
  - Data:
    - **OrderItemId (GUID, required):** Unique ID of the order item
    - **CookingEndTime (datetime, system-generated):** Timestamp when the chef completes the item
  - Validation:
    - **Order item must exist and be in "Cooking" status**
    - **Cannot mark an item as done if already served or cancelled**
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role
    - **BR-51:** The system needs to record a history of all changes made by the Manager to information products, configurations and other data
  - Functionalities:
    - Allows chefs to complete cooking by pressing the "Đã hoàn thành" butto
    - Sends real-time updates to notify waitstaff the item is ready to serve
    - Updates backend with status change and cooking end timestamp
    - Moves the item to the next phase (serving) and reflects in the UI accordingly

3.2.4.4 Done serving order item

- **Function trigger:** Triggered when a user (typically Staff or Manager) clicks the "Đã phục vụ" (Done Serving) button on an order item from the service dashboard.
- **Function description:**
  - **Roles:** Staff

- **Purpose:** Marks a specific order item as fully served to the customer. This allows tracking order fulfillment status in real-time and closing kitchen/service loops efficiently.
- **Interface:**
  - "Đã phục vụ" button is shown on each prepared dish card, typically in the service zone interface.
- **Data processing**:
  - **Input Validation**: Ensures that username, password, fullname, email, phone number, type and status are provided and valid.
  - **API Call**: Sends the staff details to the backend for storage
  - **Feedback**: Displays success or error messages based on the API response.

- **Screen layout:**

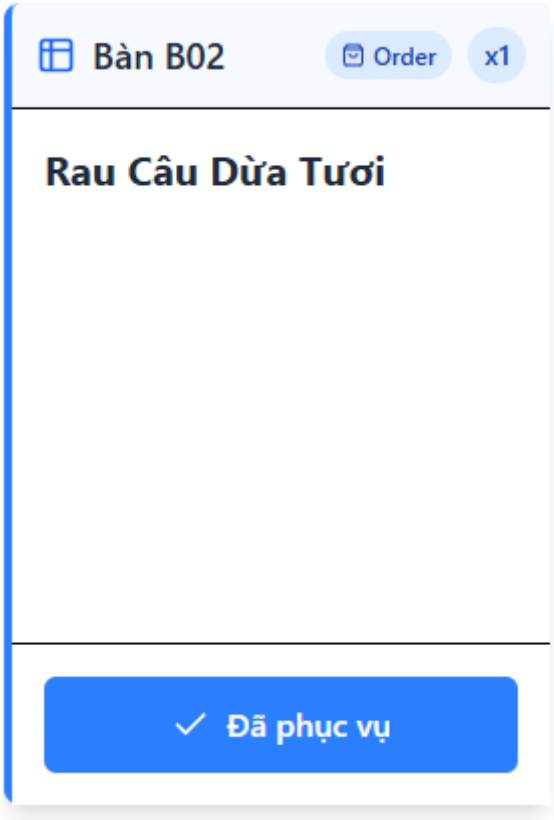


Figure 34- Done serving order item

- **Function Details:**
  - Data:
    - **OrderItemId (GUID, required):** Unique identifier of the order item being served
    - **ServeTime (DateTime, system-generated):** Timestamp when the item is confirmed as served
  - Validation:
    - **OrderItemId must exist**
    - **The order item must be in "Prepared" status (i.e., ready to be served)**
    - **Cannot serve an item that's already cancelled or completed**
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role

        - **BR-51:** The system needs to record a history of all changes made by the Manager to information products, configurations and other data
    - Functionalities:
        - Displays “Đã phục vụ” button on eligible order items
        - Once clicked, marks the item as served in the system and logs the serve time
        - Sends real-time updates to remove the item from the pending service queue
        - Triggers any necessary downstream actions (e.g., table state refresh, billing readiness)

3.2.4.5 Cancel order item

- **Function trigger:** Triggered when the user clicks the **“Hủy”** (Cancel) button on an order item from the kitchen or service interface.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows authorized users to cancel an individual dish that was previously ordered, optionally providing a cancellation reason. This helps maintain an accurate record of dish preparation and order status.
    - **Interface:**
        - Cancel button shown on each active order card
        - Optional modal to input cancellation reason
    - **Data processing**:
        - **Input Validation**: Ensures that username, password, fullname, email, phone number, type and status are provided and valid.
        - **API Call**: Sends the staff details to the backend for storage
        - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

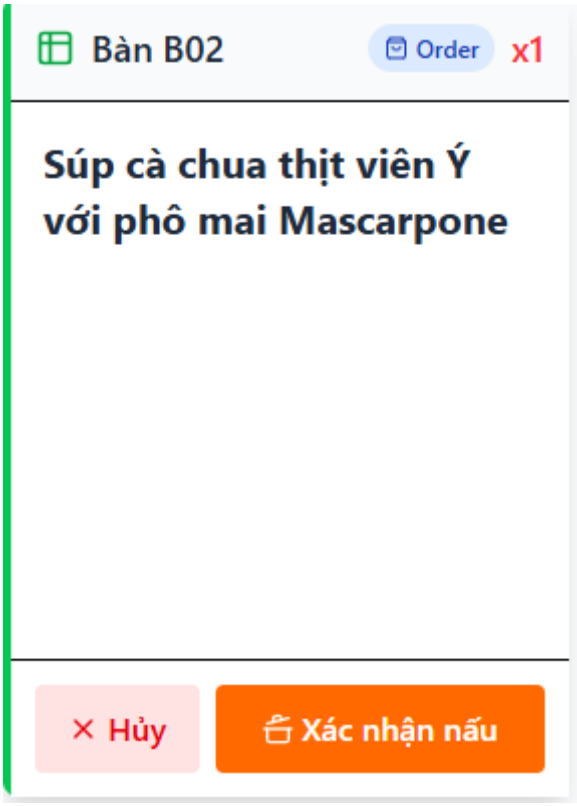


Figure 35- Cancel order item

- **Function Details:**
  - Data:
    - **OrderItemId (GUID, required):** Unique identifier of the order item
    - **ReasonCancel (string, optional):** Explanation for cancelling (e.g., customer changed mind, kitchen ran out of ingredients)
    - **EndTime (DateTime, system-generated):** Timestamp when the item was **cancelled**
  - Validation:
    - **OrderItemId must exist and be associated with an active order**
    - **Only items that are not already completed or cancelled can be cancelled again**
    - **If reason is entered, it must be no longer than 250 characters**
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role
    - **BR-51:** The system needs to record a history of all changes made by the Manager to information products, configurations and other data
  - Functionalities:
    - Displays cancel button on active order item cards
    - Optionally prompts for a reason when cancelling
    - Marks the order item as Cancelled and logs the time and reason
    - Reflects cancellation in real-time across relevant kitchen or service dashboards
    - Updates order status and dish counts accordingly

3.2.5 Staff Management

3.2.5.1 Create Staff

- **Function trigger:** This function is triggered when Manager logs in to their account, clicks on the "**Nhân viên**" button, then clicks on the "**Thêm nhân viên**" button to open the creation panel. After entering the required information, clicking the "**Thêm nhân viên**" button will create a new staff.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** To add a new staff to the system with detailed information.
  - **Interface:**
    - **Username** (Required, The username must have at least 3 characters, Password must have at least 8 characters)
    - **Password** (Required, Password must have at least 1 letter of capital, Password must have at least 1 special character)
    - **Fullname**(Required, The name must have at least 2 characters)
    - **Email** (Required, Valid email)
    - **Phone** (Required, The phone number must have at least 10 numbers)
    - **Type** (Required)
    - **Status** (Required)
  - **Data processing**:
    - **Input Validation**: Ensures that username, password, fullname, email, phone number, type and status are provided and valid.

- **API Call**: Sends the staff details to the backend for storage
- **Feedback**: Displays success or error messages based on the API response.

- **Screen layout:**

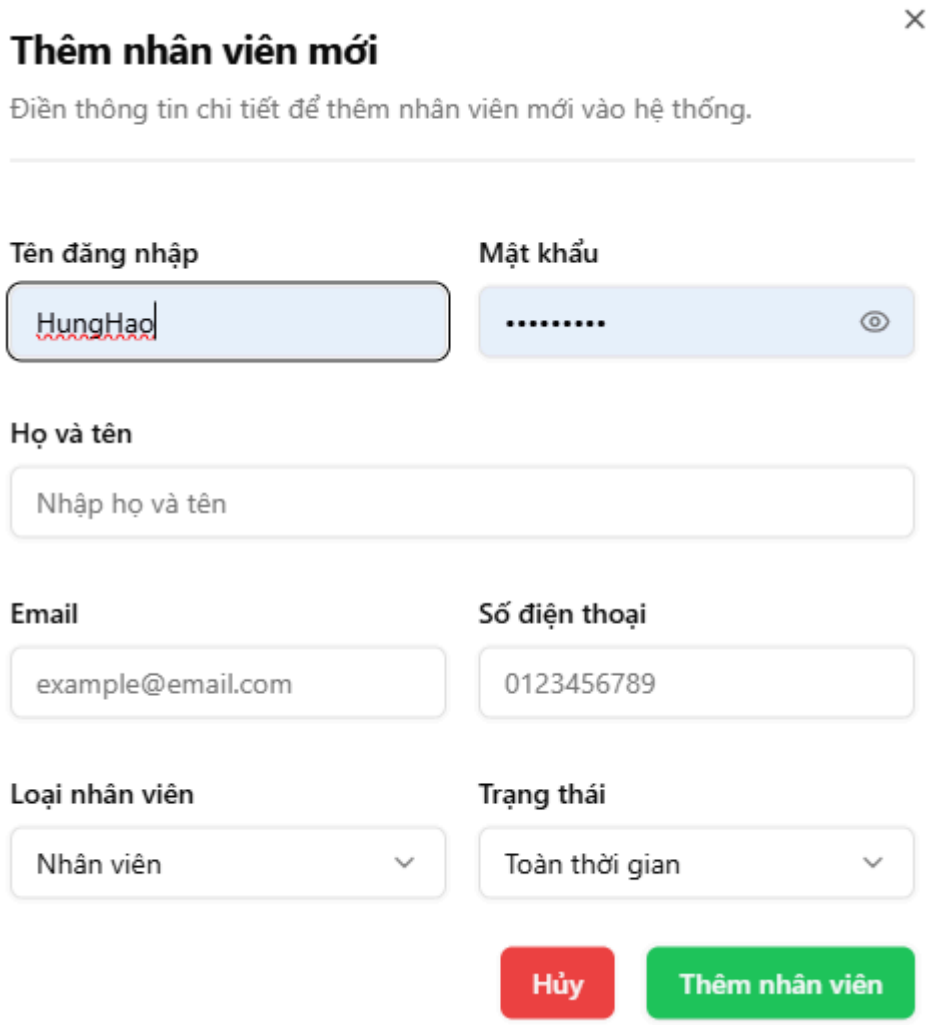


Figure 36- Create order item

- **Function Details:**
  - Data:
    - **Username** (string, required): The unique identifier for the staff.
    - **Password** (string, required): Encrypted and must meet password rules
    - **Fullname** (string, required): Full name of the staff
    - **Email** (string, required): Valid email format
    - **Phone** (string, required): Minimum 10 digits
    - **Type** (string, required): Selected from predefined staff types
    - **Status** (string, required): Set staff status full-time or part-time
  - Validation:
    - **Username:**
      - Required and must not be empty.
      - Trim whitespace.
      - Must be at least 3 characters.
    - **Password:**
      - Required and must not be empty.

        - Must be at least 8 characters.
        - Must contain at least 1 uppercase letter.
        - Must contain at least 1 special character.
    - **Full Name:**
        - Required and must not be empty.
        - Must have at least 2 characters.
    - **Email:**
        - Required and must match a valid email pattern.
    - **Phone Number:**
        - Required and must contain at least 10 digits.
        - Must only contain numbers.
    - **Staff Type:**
        - Required; must select from the predefined list
    - **Status:**
        - Required; must select from the predefined list.
- Business rules:
    - **BR-03:** Manager can create, edit, and delete Staff and Chef accounts.
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-51:** The system needs to record a history of all changes made by the Manager to information products, configurations and other data.
    - **BR-53:** When creating a staff account, the system must validate that all input fields meet format and security requirements (e.g., password strength, valid email/phone number, username uniqueness), and reject invalid input.
- Functionalities:
    - Validates all input fields when submitting the form
    - Prevents form submission if any required field is empty or invalid
    - Sends the validated data to the backend using axios.POST
    - Handles the API response:
        - Shows a success message upon successful creation
        - Displays appropriate error messages if an error occurs
    - Resets the form fields after a successful staff creation

3.2.5.2 View Employee List

- **Function trigger:** Triggered when the Manager logs into their account and clicks on the "Nhân viên" button in the sidebar. The system loads and displays the list of staff and chef.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows Manager to view all registered members, search by name/email/phone, filter by staff type and working status, and navigate through the list with pagination.
    - **Interface:**
        - Search input: text field for searching by name, email, or phone
        - Filter by Status: dropdown (e.g., Toàn thời gian, Bán thời gian)
        - Filter by Type: dropdown (e.g., Nhân viên, Quản lý, Đầu bếp)

        - Staff table with columns
        - Pagination controls: hiển thị số dòng và chuyển trang
    - **Data processing**:
        - **Input Validation**:
            - Search input: allow partial matching, case-insensitive
            - Status & Type filters: must match predefined values
        - **API Call**: Sends a request to get the list of staff from the backend
        - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

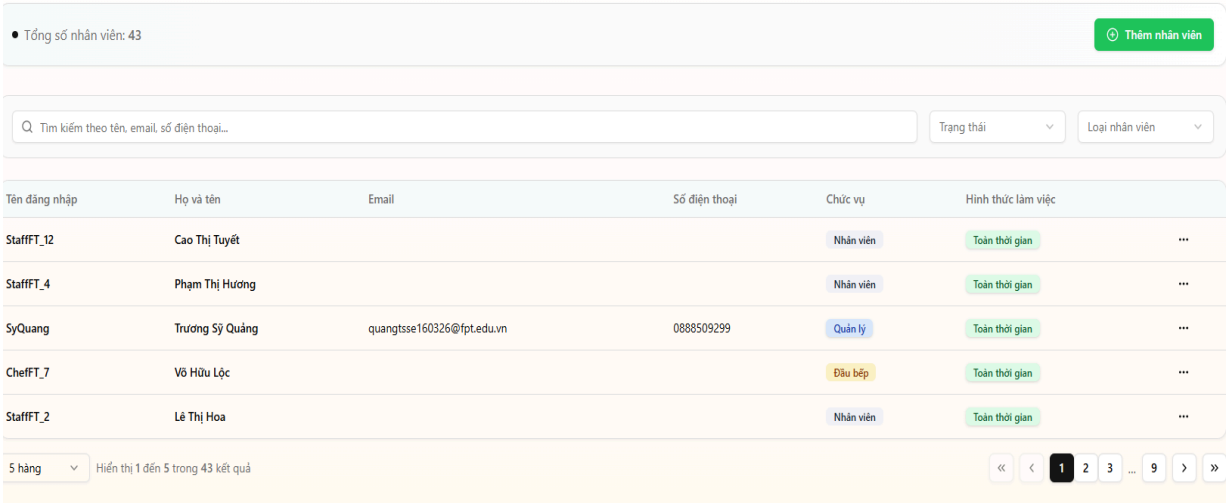


Figure 37- View Employee List

- **Function Details:**
    - Data:
    - Validation:
    - Business rules:
        - **BR-03:** Manager can create, edit, and delete Staff and Chef accounts.
        - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
        - **BR-51:** The system needs to record a history of all changes made by the Manager to information products, configurations and other data.
    - Functionalities:
        - Loads staff list on page visit
        - Allows searching staff by name/email/phone
        - Allows filtering staff by working status and staff type
        - Supports pagination with page indicators (e.g., 1 2 3 … 9)
        - Updates UI instantly when filters or search terms are changed
        - Displays total staff count at the top

3.2.5.3 Delete Employee

- **Function trigger:** Triggered when the Manager logs into their account and clicks on the three-dot menu (**...**) at the end of a staff row in the staff list and selects the "**Xóa**" option. A confirmation modal will appear, requiring the Manager to confirm the deletion.

- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows the Manager to remove a staff member from the system when they are no longer active or valid.
  - **Interface:** Confirmation Dialog
  - **Data processing**:
    - **Input Validation**:
      - Ensure the staff ID exists before sending the request
    - **API Call**: Sends a request to delete staff from the backend
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

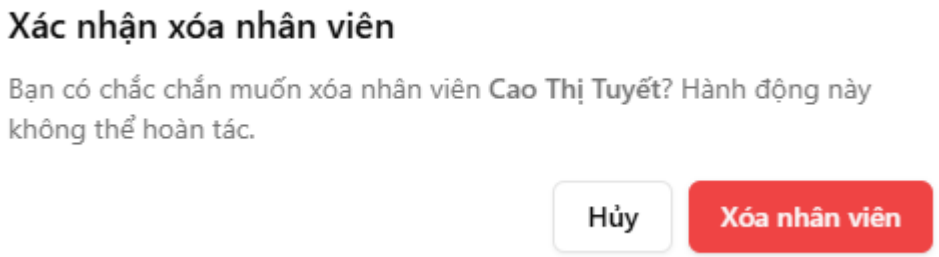


Figure 38- Delete Employee

- **Function Details:**
  - Data:
    - staffId (string, required): The unique identifier of the staff to be deleted
  - Validation:
    - staffId must be valid and exist
  - Business rules:
    - **BR-03:** Manager can create, edit, and delete Staff and Chef accounts.
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-51:** The system needs to record a history of all changes made by the Manager to information products, configurations and other data.
  - Functionalities:
    - Show confirmation modal → On **Confirm**, trigger API
    - Displays success or error messages depend on the API response
    - Refresh only the table data, not the entire page

3.2.6 Working Slot Management

3.2.6.1 Create working slot

- **Function trigger:** This function is triggered when Manager logs in to their account, clicks on the "**Lịch làm việc**" button, then clicks on the "**Tạo mới**" button to open the creation panel. On the interface, a two-page form will appear. The first page is for creating the work **shift** (ca làm) and the second page is for entering the **working slot** information (lịch làm việc).
- **Function description:**
  - **Roles:** Manager

- **Purpose:** To add a new shift and working slot to the system with detailed information.
- **Interface:**
  - **Shift Name** (Required)
  - **Description**
  - **Shift Id**(Required)
  - **Day of Week** (Required)
  - **Shift Start** (Required)
  - **Shift End** (Required)
  - **Part-time Capacity**
- **Data processing**:
  - **Input Validation**: Ensures that shift name, description, shift, day, shift start, shift end and number of part-time employees are provided and valid.
  - **API Call**: Sends the shift and working slot details to the backend for storage
  - **Feedback**: Displays success or error messages based on the API response.

- **Screen layout:**

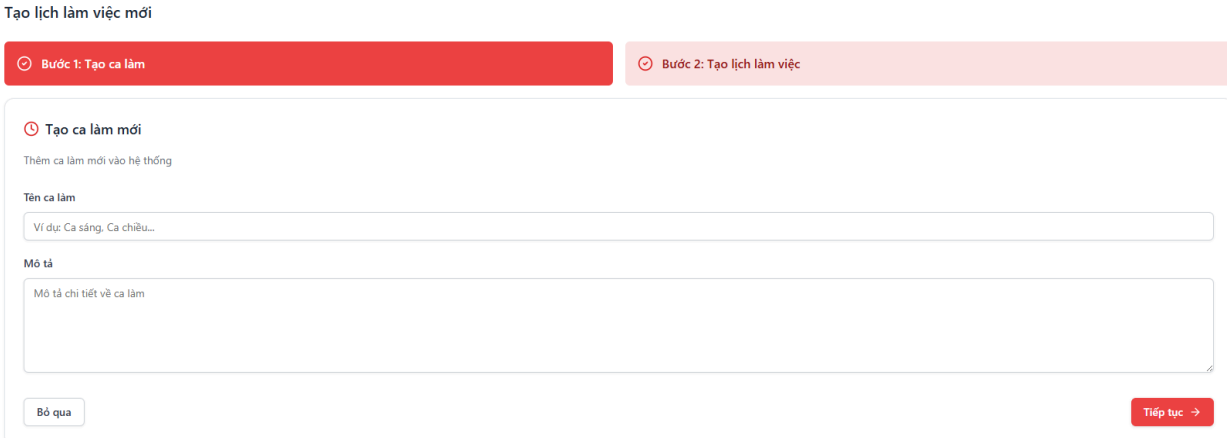


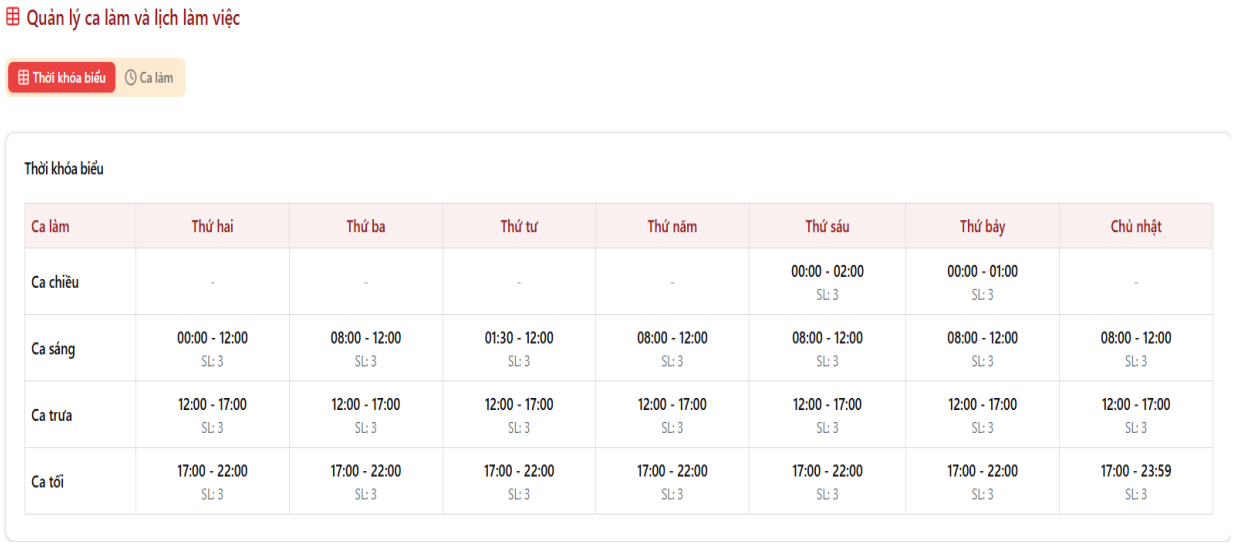
Figure 39- Create working slot

- **Function Details:**
  - Data:
    - **Shift Name** (string, required): The unique identifier for the staff
    - **Description** (string, required): Encrypted and must meet password rules
    - **Shift Id** (string, required): Full name of the staff
    - **Day of Week** (enum, required): Valid email format
    - **Shift Start** (time, required): Minimum 10 digits
    - **Shift End** (time, required): Selected from predefined staff types
    - **Part-Time Staff Capacity** (int, required): Set staff status full-time or part-time
  - Validation:
    - **Shift Name:**
      - Required and must not be empty
      - Must have at least 3 characters
      - Trim whitespace
    - **Shift Type:**
      - Required
      - Must be selected from predefined list
    - **Day of Week:**
      - Required
      - Must be a valid date, not in the past
    - **Shift Start & Shift End:**

        - Both required
        - Start time must be earlier than end time
    - **Part-Time Staff Capacity:**
        - Must be a non-negative integer
- Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-16:** Each full-time staff member is assigned to one zone (kitchen or dining) for the whole week.
    - **BR-17:** Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.
    - **BR-18:** Full-time chefs must be assigned to a kitchen area. If no kitchen zone is available, the system must reject the assignment.
    - **BR-25:** Registrations must be made within a 2 weeks range.
    - **BR-26:** A staff member can only register for up to 3 shifts per week.
    - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
- Functionalities:
    - Renders a multi-step form to input work shift and working slot details
    - Performs validation on all required fields during submission
    - Prevents submission if data is invalid or incomplete
    - Sends valid data to backend via API (axios.POST)
    - Displays appropriate success/error messages depending on API response
    - Resets the form after a successful creation
    - Allows going back to the previous step for editing before submission

3.2.6.2 View working slot

- **Function trigger:** This function is triggered when the Manager logs into their account, clicks on the "**Lịch làm việc**" button then clicks "**Quản lý**" button. The system displays a list of all existing working slots.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows the Manager to view all created working slots including their details. No edit or delete actions are available.
  - **Interface:** Working slot list view
  - **Data processing**:
    - **Input Validation**:
    - **API Call**: Sends a request to get the list of staff from the backend
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**



Ca làm	Thứ hai	Thứ ba	Thứ tư	Thứ năm	Thứ sáu	Thứ bảy	Chủ nhật
Ca chiều	-	-	-	-	00:00 - 02:00 SL: 3	00:00 - 01:00 SL: 3	-
Ca sáng	00:00 - 12:00 SL: 3	08:00 - 12:00 SL: 3	01:30 - 12:00 SL: 3	08:00 - 12:00 SL: 3	08:00 - 12:00 SL: 3	08:00 - 12:00 SL: 3	08:00 - 12:00 SL: 3
Ca trưa	12:00 - 17:00 SL: 3	12:00 - 17:00 SL: 3	12:00 - 17:00 SL: 3	12:00 - 17:00 SL: 3	12:00 - 17:00 SL: 3	12:00 - 17:00 SL: 3	12:00 - 17:00 SL: 3
Ca tối	17:00 - 22:00 SL: 3	17:00 - 22:00 SL: 3	17:00 - 22:00 SL: 3	17:00 - 22:00 SL: 3	17:00 - 22:00 SL: 3	17:00 - 22:00 SL: 3	17:00 - 23:59 SL: 3

Figure 40- View working slot

- **Function Details:**
  - Data: Available working slots
  - Validation:
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
  - Functionalities:
    - Display available working slots details
    - Uses (axios.GET) to retrieve working slot data from the backend

3.2.6.3 View shift

- **Function trigger:** This function is triggered when the Manager logs into their account, clicks on the "**Lịch làm việc**" button, and navigates to the "**Ca làm**" tab.

- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows the Manager to view the list of predefined work shifts in the system. Each shift can be reviewed but not created or deleted from this screen.
  - **Interface:** Shift list view
  - **Data processing**:
    - **Input Validation**:
    - **API Call**: Sends an (axios.GET) request to fetch the list of shifts from the backend
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

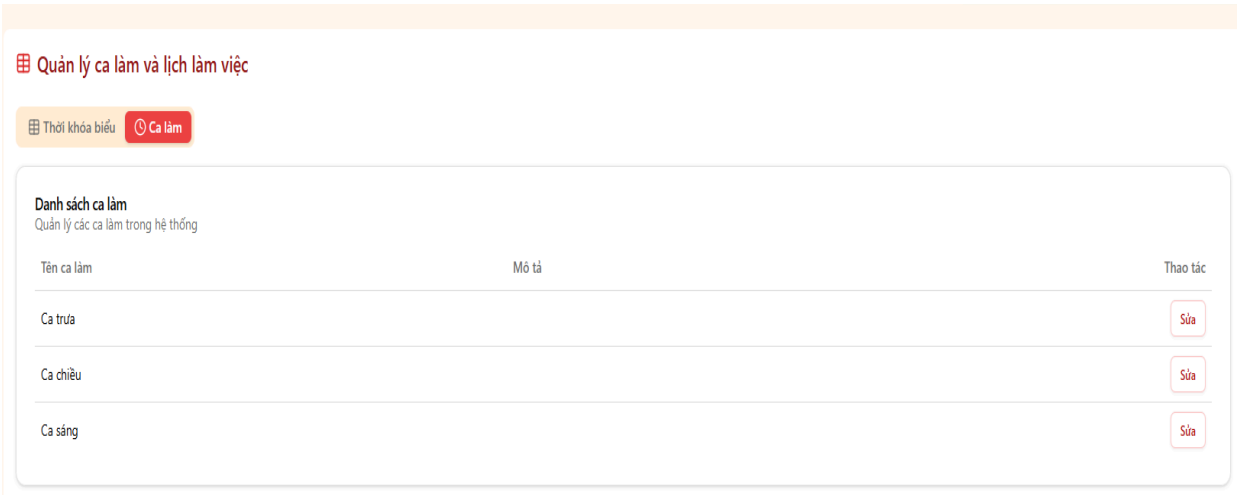


Figure 41- View shift

- **Function Details:**
  - Data: Available shifts
  - Validation:
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
  - Functionalities:
    - Fetch and display the list of shifts from backend
    - Show shift details clearly in a table format

3.2.6.4 Edit shift information

- **Function trigger:** This function is triggered when the Manager logs into their account, clicks the "**Lịch làm việc**" button, then goes to the "**Ca làm**" tab and clicks the "**Sửa**" button.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows the Manager to edit basic information of an existing shift, specifically the shift name and description.

- **Interface:**
  - Shift Name (required)
  - Description (optional)
- **Data processing**:
  - **Input Validation**:
  - **API Call**: Sends an (axios.PUT) request to fetch the list of shifts from the backend
  - **Feedback**: Displays success or error messages based on the API response.

- **Screen layout:**

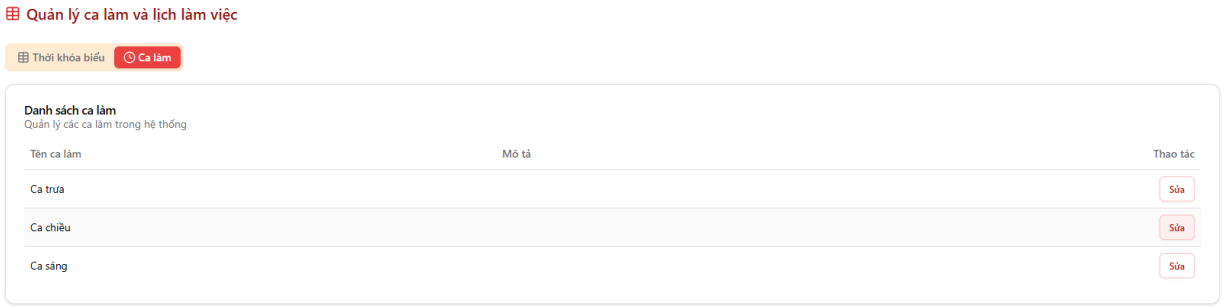


Figure 42- Create working slot

- **Function Details:**
  - Data:
    - Shift Name (string, required): New or updated name for the shift
    - Description (string, optional): Additional details about the shift
  - Validation:
    - Shift name must not be empty and must have at least 3 characters
    - Both fields should remove leading/trailing whitespace before submission
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-51:** The system needs to record a history of all changes made by the Manager to information products, configurations and other data.
    - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
  - Functionalities:
    - Allows Manager to open and edit a shift's name and description
    - Validates inputs before sending data
    - Sends the updated data to the backend
    - Shows appropriate feedback messages
    - Refreshes the shift list after a successful update
    - Cancels edit without saving if the user closes the form or clicks "Hủy"

3.2.7 Staff Working Slot Register Management

3.2.7.1 Register working slot

- **Function trigger:** This function is triggered when a part-time employee logs in to their account, clicks on the "Cá nhân" button, then clicks on the "Đăng ký giờ làm việc" button to open the panel. After selecting the working slot, clicking the "Xác nhận đăng ký" button will register the working slot.
- **Function description:**
  - **Roles:** Staff,Chef
  - **Purpose:** Part-time staff are allowed to register their available working slots in the system.
  - **Interface:**
  - **Data processing**:
    - **Input Validation**:
    - **API Call**: Sends the registered working slot details to the backend for storage
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

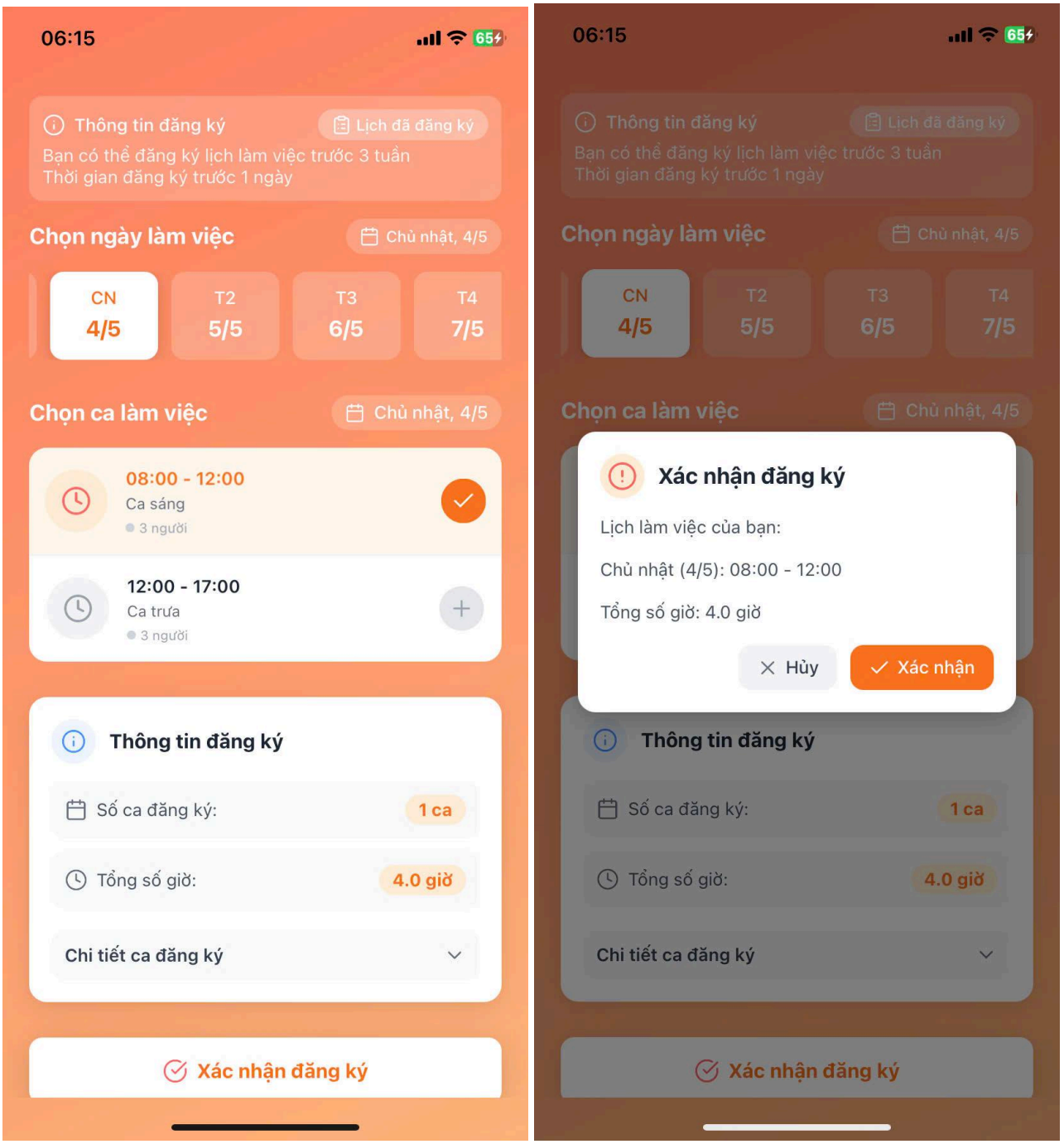


Figure 43- Register working slot

- **Function Details:**
  - Data:
    - **Zone** (string, required): Selected from predefined staff types
  - Validation: Users are required to choose a valid working slot
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-13:** A staff member can only be scheduled for a shift if they have previously registered for that working slot on that specific day.

        - **BR-25:** Registrations must be made within a 2 weeks range.
        - **BR-26:** A staff member can only register for up to 3 shifts per week.
        - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
    - Functionalities:
        - Displays a multi-step form to view and register available working slots
        - Validates all required fields before allowing submission
        - Prevents submission if any field is invalid or incomplete
        - Sends valid data to backend via API (axios.POST)
        - Displays success or error message depending on API response
        - Resets the form after a successful registration
        - Allows users to return to the previous step to modify selections

3.2.8 Staff Zone Schedules Management

3.2.8.1 Create Staff Zone Schedule

- **Function trigger:** This function is triggered when Manager logs into their account, clicks on the "**Lịch làm việc**" button, then clicks on the calendar icon button to open the details. Click the "**Yêu cầu đăng ký**" tab. After selecting the staff's working slot register and to assign working zone, clicking the "**Phân công khu vực**" button will create a staff zone schedule.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows Manager to assign a zone to a staff's registered working slot. Creates a staff zone schedule upon confirmation.
    - **Interface:**
        - **Zone** (Required)
    - **Data processing**:
        - **Input Validation**: Ensures that zone are provided
        - **API Call**: Sends the staff zone schedule details to the backend for storage
        - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

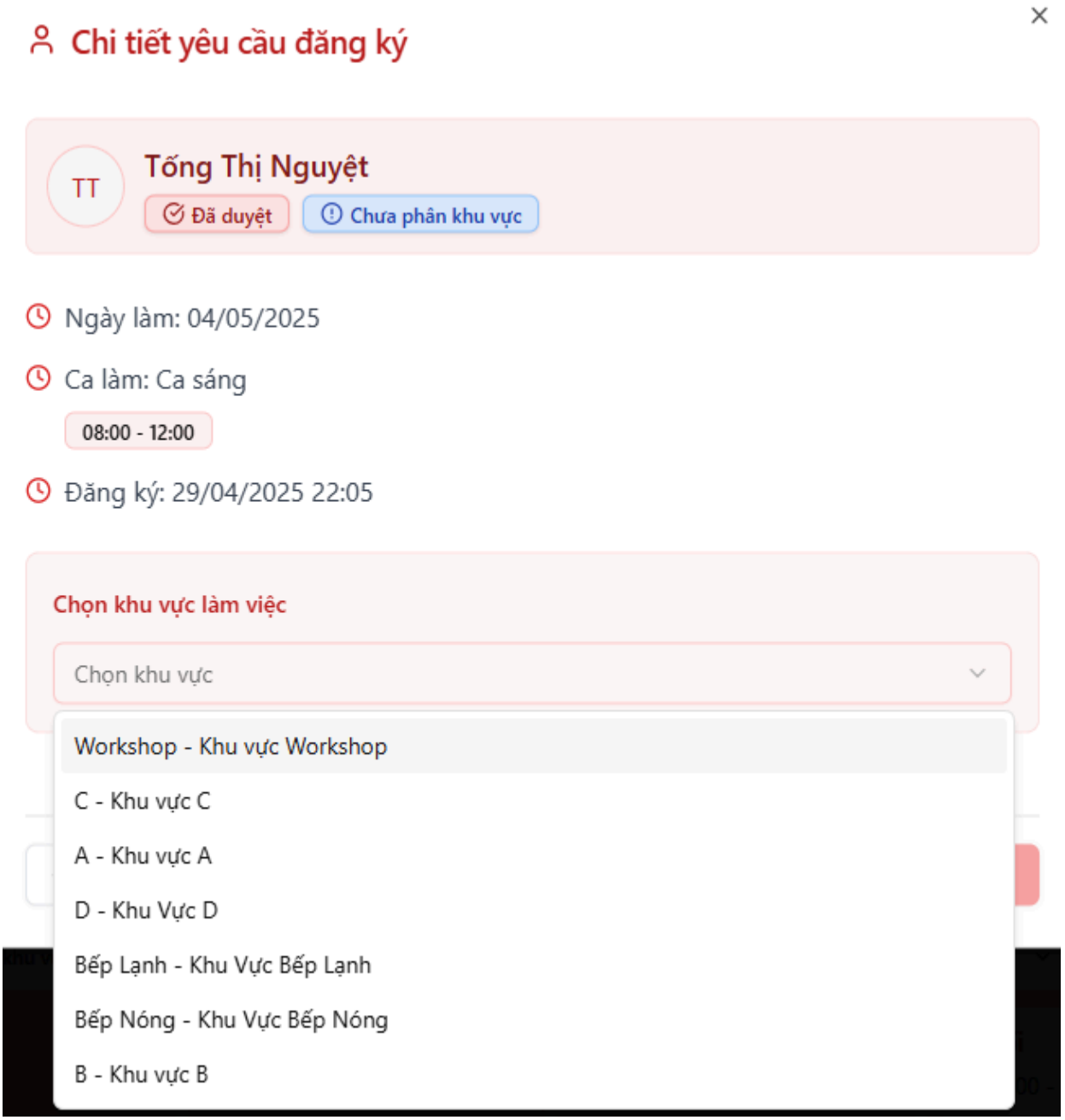


Figure 44- Create Staff Zone Schedule

- **Function Details:**
  - Data:
    - **ZoneId** (string, required): Selected from predefined zones
  - Validation:
    - **Zone:**
      - Required; must select from the predefined list.
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-14:** A staff member can only be scheduled to one zone per working date and working slot.

        - **BR-17:** Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.

        - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
    - Functionalities:
        - Displays a list of registered working slots in the "Yêu cầu đăng ký" tab.
        - Allows Manager to select a staff registration and choose a zone from a predefined list.
        - Validates that a zone has been selected before allowing submission.
        - Sends valid data to backend via API (axios.POST)
        - Displays success or error messages based on the API response.
        - Prevents duplicate assignments for the same time slot.
        - Updates the interface and resets selection after successful assignment.

3.2.8.2 View Staff Zone Schedule

- **Function trigger:** This function is triggered when the Manager navigates to the "**Lịch làm việc**" screen, selecting a specific week.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows the Manager to view assigned working zones and the staff members assigned to each zone per shift..
    - **Interface:**
        - Weekly/Monthly calendar layout.
        - “View Zones” button under each shift opens a detailed view listing assigned zones and staff.
    - **Data processing**:
        - **Input Validation**:
        - **API Call**: Sends the GET api to get the details of staff zone schedule from the backend
        - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

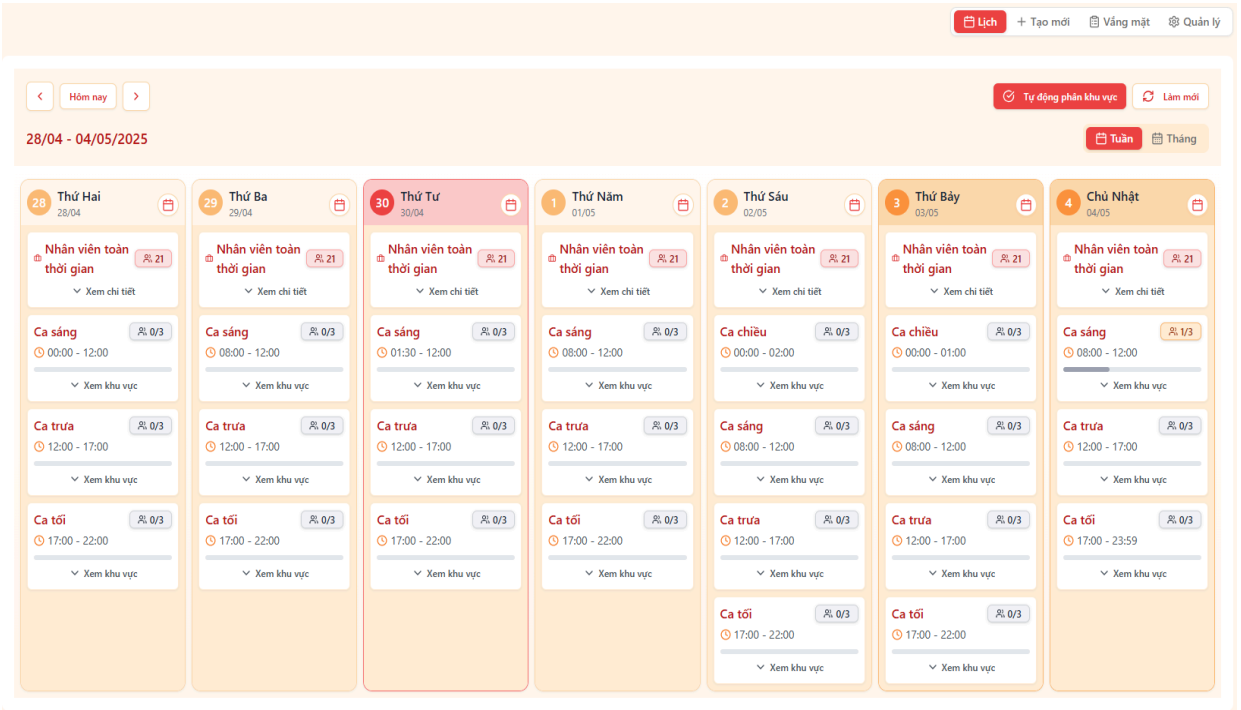


Figure 45- View Staff Zone Schedule

- **Function Details:**
  - Data:
    - WorkingDate (DateOnly): The specific date on which the staff is scheduled to work.
    - WorkingSlotId (GUID): Unique identifier of the working shift assigned to the staff.
    - WorkingSlot (object): Detailed information about the working shift.
    - DayOfWeek (string): The day of the week corresponding to the working date.
    - ZoneId (GUID): Unique identifier of the zone assigned for the working shift.
    - ZoneName (string): Name of the assigned working zone.
    - StaffId (GUID): Unique identifier of the staff member assigned to the zone.
    - StaffName (string): Full name of the assigned staff member.
  - Validation:
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-14:** A staff member can only be scheduled to one zone per working date and working slot.
    - **BR-17:** Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.
    - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
  - Functionalities:
    - Display weekly/monthly work schedule.

- Show total staff count per shift.
- "View Zones" reveals:
  - List of zones assigned for that shift.
  - Staff names under each zone.
- Pagination or scroll if the staff list is long.

3.2.9 Staff Zone Management

3.2.9.1 Synchronize Staff Zone

- **Function trigger:** This function is triggered when Manager logs into their account, clicks on the "**Khu vực nhân viên**" button, then click on the "**Xem lịch phân công**" button.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Synchronize staff by area and shift in real time.
  - **Interface:**
    Data processing:
    - **Input Validation**:
    - **API Call**: Sends the staff zone details to the backend for storage
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

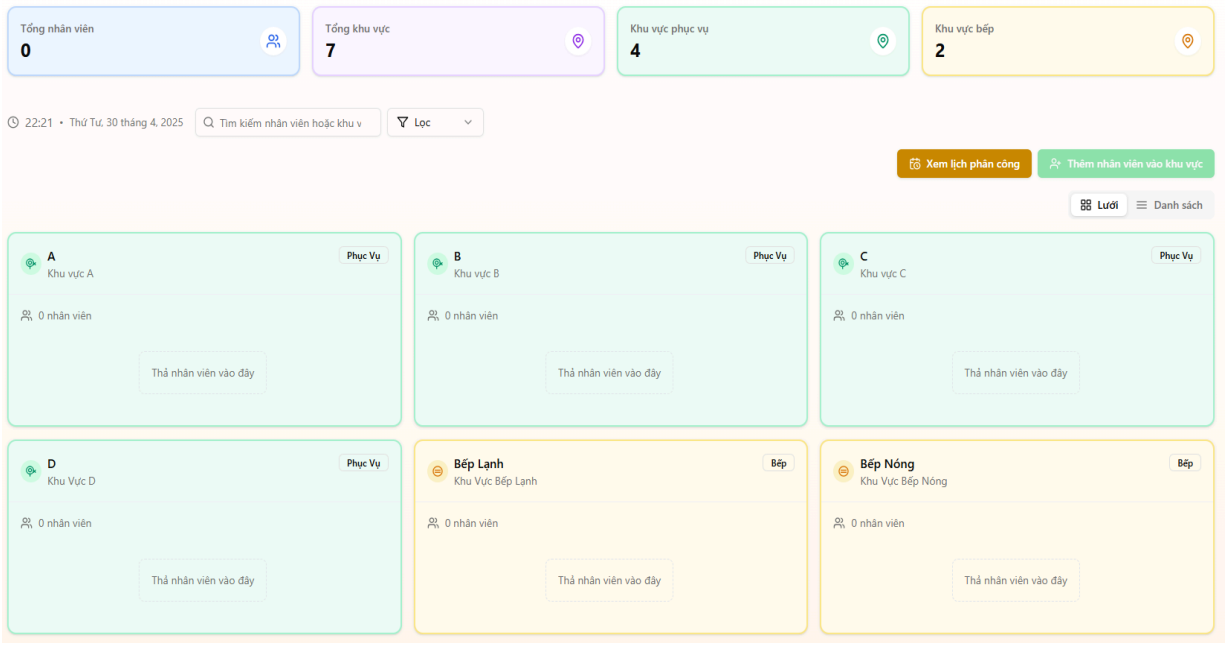


Figure 46- Synchronize Staff Zone

- **Function Details:**
  - Data:
  - Validation:
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-13:** A staff member can only be scheduled for a shift if they have previously registered for that working slot on that specific day.

        - **BR-14:** A staff member can only be scheduled to one zone per working date and working slot.
        - **BR-17:** Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.
        - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
    - Functionalities:
        - Retrieves the list of staff with already assigned working slots.
        - Synchronizes staff zones based on their assigned working schedule.
        - Excludes staff who have been marked as absent.
        - Does not overwrite zones that were manually assigned previously.
        - Sends valid data to backend via API (axios.POST)
        - Displays success or error messages based on the API response.
        - Updates the staff zone schedule view after a successful synchronization.

3.2.9.2 View Staff Zone

- **Function trigger:** This function is triggered when the Manager logs into their account, clicks on the "**Khu vực nhân viên**" button.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** To view the current assignment of staff members to specific zones by department and shift.
    - **Interface:** Staff zone list view
      Data processing:
        - **Input Validation**:
        - **API Call**: Sends a request to get the list of staff from the backend
        - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

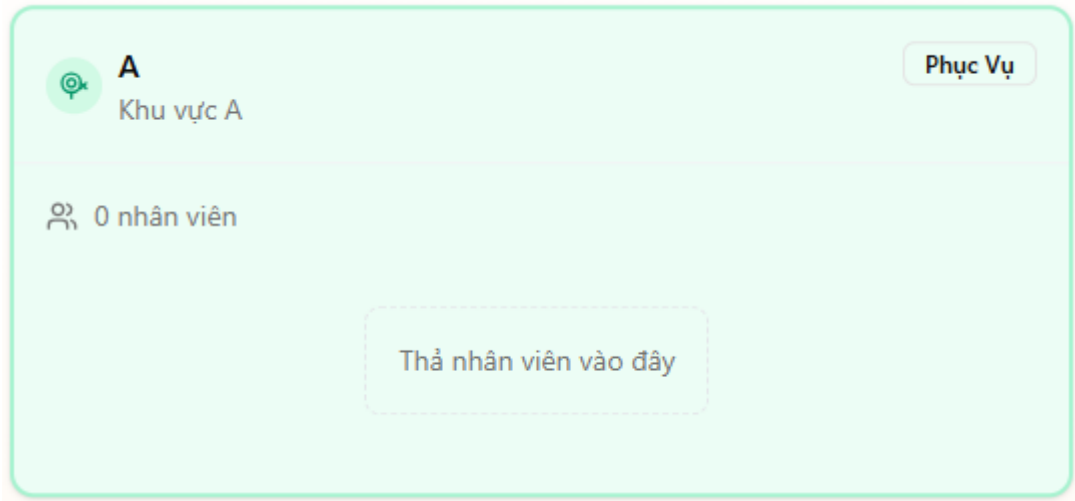


Figure 47- View Staff Zone

- **Function Details:**
    - Data:

        - Staff ID (string, Required): The unique identifier of the staff
        - Staff Name (string, Required): The full name of the staff member
        - Type (string, Required): Staff role (Staff, Chef)
        - Type (string, Required): Type of employment (Full-time, Part-time)
        - Zone ID (string, Required): The unique identifier of the assigned zone
        - Zone Name (string, Required): The display name of the assigned zone
        - Zone Type (string, Required): The type of zone (Service, Kitchen)
    - Validation:
    - Business rules:
        - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
        - **BR-13:** A staff member can only be scheduled for a shift if they have previously registered for that working slot on that specific day.
        - **BR-14:** A staff member can only be scheduled to one zone per working date and working slot.
        - **BR-17:** Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.
        - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
    - Functionalities:
        - Retrieves and renders real-time staff-zone assignment data
        - Allows users to switch between list and grid layouts
        - Filters and searches staff/zone dynamically
        - Staff details are clearly categorized by zone and zone type
        - Excludes absent staff automatically from display

3.2.9.3 Reassign Staff to Zone via Drag and Drop

- **Function trigger:** Triggered when Manager clicks on the "**Khu vực nhân viên**" page and drags a staff card from one zone to another drop area.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows managers to reassign a staff member from one zone to another via drag-and-drop.
    - **Interface:**
        - Staff list by zone shown in grid or list layout
        - Each staff member is displayed as a draggable card
        - Zones are represented as droppable containers

      Data processing:
        - **Input Validation**:
        - **API Call**: Sends a request to update staff from the backend
        - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

Figure 48- Reassign Staff to Zone

- **Function Details:**
  - Data:
    - StaffID (string, Required): The unique identifier of the staff
    - ZoneID (string, Required): The zone to which the staff is being reassigned
  - Validation:
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-14:** A staff member can only be scheduled to one zone per working date and working slot.
    - **BR-16:** Each full-time staff member is assigned to one zone (kitchen or dining) for the whole week.
    - **BR-17:** Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.
    - **BR-18:** Full-time chefs must be assigned to a kitchen area. If no kitchen zone is available, the system must reject the assignment.
  - Functionalities:
    - Enables drag-and-drop from one zone to another
    - On drop, triggers an API call (axios.PUT) to update staff's zone
    - Validates role compatibility with target zone
    - Displays success or error messages based on API response
    - Refreshes UI to reflect the updated staff distribution

3.2.9.4 Add Staff to Zone

- **Function trigger:** This function is triggered when the Manager logs into their account, navigates to "**Khu vực nhân viên**", and clicks on the "**Thêm nhân viên vào khu vực**" button. A dialog appears allowing selection of a target zone and eligible staff.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Assign staff to a specific work zone
  - **Interface:**
    - Dropdown: Select Zone
    - Dropdown: Select Zone Type (Dining or Kitchen)
    - Staff List: Full name, role, employment type (full-time or part-time)

    Data processing:

        - **Input Validation**:
            - Validates that a zone is selected
            - Ensures that the selected staff is not already assigned to any zone
            - Checks that staff's role matches the target zone type (chef → kitchen)
            - Blocks adding absent staff for the current day
        - **API Call**: Sends a request to create staffzone from the backend
        - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

Figure 49- Add Staff to Zone

- **Function Details:**
    - Data:
        - StaffID (string, Required): The unique identifier of the staff
        - ZoneID (string, Required): The zone to which the staff is being assigned
    - Validation:
    - Business rules:
        - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
        - **BR-14:** A staff member can only be scheduled to one zone per working date and working slot.
        - **BR-16:** Each full-time staff member is assigned to one zone (kitchen or dining) for the whole week.
        - **BR-17:** Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.
        - **BR-18:** Full-time chefs must be assigned to a kitchen area. If no kitchen zone is available, the system must reject the assignment.
        - **BR-19:** Full-time waiters or servers (non-chefs) must be assigned to a dining area. If no dining zone is available, the system must reject the assignment.
        - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
    - Functionalities:
        - Staff must exist in the system and not be marked as absent on the assigned date
        - Zone must be a valid, existing zone
        - Staff must not already belong to another zone in the same shift
        - Staff role must match the zone type (Staff to dining zone, Chef to Kitchen zone)

3.2.9.5 Remove Staff from Zone via Drag-and-Drop

- **Function trigger:** This function is triggered when the Manager logs into their account, navigates to "**Khu vực nhân viên**", and drags a staff member's card and drops it onto a delete icon in the Staff Zone Management interface.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** To allow the manager to remove a staff member from their currently assigned zone via an intuitive drag-and-drop interaction.
  - **Interface:**
    - Staff cards are draggable
    - A trash/delete icon is visible in the UI
    - Dropping a card onto the delete icon triggers the removal

    Data processing:
    - **Input Validation**:
    - **API Call**: Sends a request to delete staffzone from the backend
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

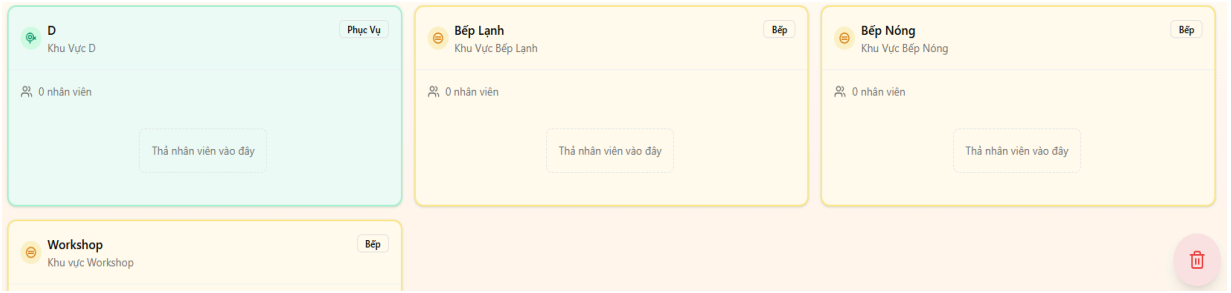


Figure 50- Remove Staff from Zone

- **Function Details:**
  - Data:
    - StaffID (string, Required): The unique identifier of the staff
    - ZoneID (string, Required): The zone to which the staff is being assigned
  - Validation:
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-14:** A staff member can only be scheduled to one zone per working date and working slot.
    - **BR-16:** Each full-time staff member is assigned to one zone (kitchen or dining) for the whole week.
    - **BR-17:** Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.
    - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.

- Functionalities:
  - Supports drag-and-drop interaction to the delete icon
  - Refreshes the zone view to remove the staff from the displayed list

3.2.10 Absent Management

3.2.10.1 Create Absent

- **Function trigger:** This function is triggered when Manager logs into their account, clicks on the "**Lịch làm việc**" button, then click on the "**Vắng mặt**" button to open. Click the **"Thêm mới"** to open the creation panel. After entering the required information, clicking the "**Lưu**" button will create a new absence.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows Manager to mark a staff's absence by shift or full day. Creates an absence record upon confirmation.
  - **Interface:**
    - **Staff** (Required)
    - **Absence Day** (Required)
    - **Full-Day Absence** (if not taking a specific working slot)
    - **Working Slot** (if not taking a full-day absence)
  - **Data processing**:
    - **Input Validation**: Ensures that Staff, Absence day and working slot are provided and valid
    - **API Call**: Sends the employee absence to the backend for storage
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

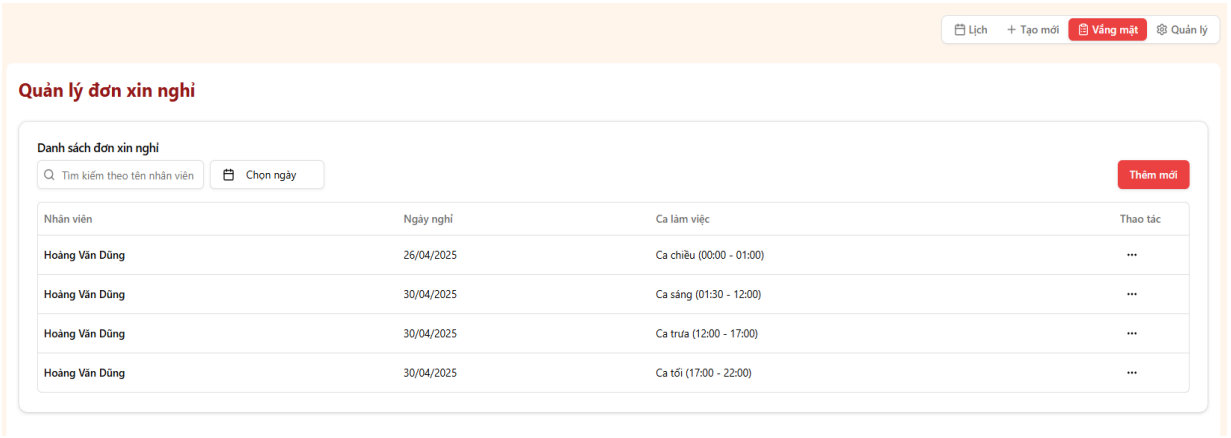


Figure 51- Create Absent

- **Function Details:**
  - Data:
    - Staff (string, required): Selected from the list of staff.
    - Absence Day (date, required): The date on which the staff is absent.
    - Full-Day Absence (boolean, optional): If checked, the absence applies for the entire day.
    - Working Slot (object, optional): Selected if not a full-day absence (shift-based absence).
  - Validation:
    - Staff: Required; must be selected from the existing staff list.

        - Absence Day: Required; must be a valid date and not in the past.
        - Working Slot: Required if "Full-Day Absence" is not selected.
        - Cannot mark both full-day and specific slot simultaneously.
    - Business rules:
        - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
        - **BR-13:** A staff member can only be scheduled for a shift if they have previously registered for that working slot on that specific day.
        - **BR-14:** A staff member can only be scheduled to one zone per working date and working slot.
        - **BR-16:** Each full-time staff member is assigned to one zone (kitchen or dining) for the whole week.
        - **BR-17:** Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.
        - **BR-54:** Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.
        - **BR-55:** The system must prevent assigning a zone to any staff member marked as absent for a given working date and slot. Absence records must block zone assignment and be mutually exclusive with existing shift assignments.
        -
    - Functionalities:
        - Displays a searchable list of absences by staff name and date.
        - Allows Manager to add a new absence by selecting staff, date, and either a full day or a specific working slot.
        - Validates required fields before submission.
        - Sends valid data to the backend via API (axios.POST).
        - Displays success or error messages depending on the API response.
        - Allows deletion of existing absences.
        - Refreshes the list and clears form upon successful submission.

3.2.10.2 View Employee Absent

- **Function trigger:** This function is triggered when Manager logs into their account, clicks on the "**Lịch làm việc**" button, then click on the "**Vắng mặt**" button to view the absence list.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows Manager to view all staff absence records, filter by staff name and date, and manage.
    - **Interface:**
        - Search by staff name (text input)
        - Filter by absence date (date picker)
        - Absence list with columns: Staff Name, Absence Date, Working Slot
    - **Data processing**:

- **Input Validation**: Validates optional search inputs (e.g., valid date format for filters).
- **API Call**: Sends a request to get the list of staff absences from the backend
- **Feedback**: Displays success or error messages based on the API response.

- **Screen layout:**

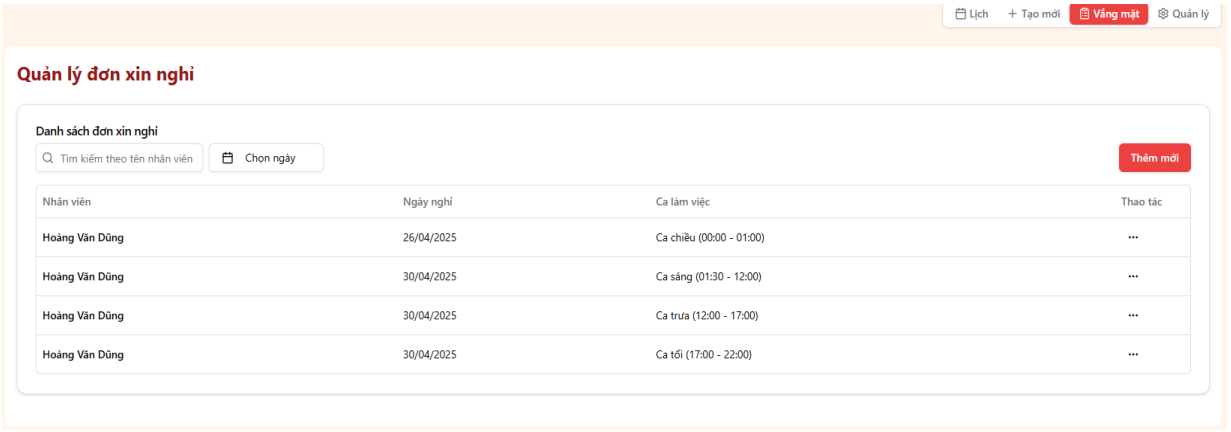


Figure 52- View Employee Absent

- **Function Details:**
  - Data:
  - Validation:
  - Business rules:
    - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - **BR-13:** A staff member can only be scheduled for a shift if they have previously registered for that working slot on that specific day.
    - **BR-14:** A staff member can only be scheduled to one zone per working date and working slot.
    - **BR-17:** Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.
    - **BR-55:** The system must prevent assigning a zone to any staff member marked as absent for a given working date and slot. Absence records must block zone assignment and be mutually exclusive with existing shift assignments.
    -
  - Functionalities:
    - Get list of data to the backend via API (axios.GET).
    - Displays a searchable and filterable list of staff absences.
    - Allows filtering the list by staff name and absence date.
    - Shows clear labels for full-day or shift-based absences.

3.2.10.3 Delete Employee Absent

- **Function trigger:** This function is triggered when Manager logs into their account, clicks on the "**Lịch làm việc**" button, then click on the "**Vắng mặt**" button to view the absence list. After selecting the delete option (“**Xóa**”) from the action menu (“**...**”) of a specific absence record and confirm the appear dialog before deletion.

- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows Manager to delete an existing staff absence record after confirmation.
  - **Interface:**
    - Action menu (“**...**”) for each absence entry
    - Confirmation dialog: “**Xác nhận xóa**”
    - “**Xác nhận**” and “**Hủy**” buttons
  - **Data processing**:
    - **Input Validation**: Ensures the selected absence ID exists and is valid.
    - **API Call**: Sends a request to delete staff absences from the backend
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

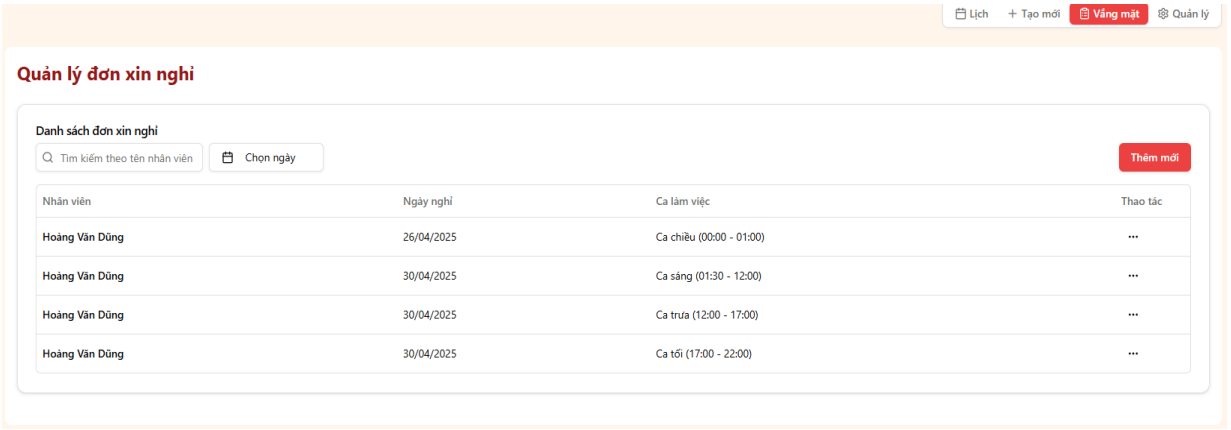


Figure 53- Delete Employee Absent

- **Function Details:**
  - Data:
    - Absence ID (GUID): Unique identifier of the absence record to be deleted
  - Validation: Ensures the selected absence ID exists and is valid.
  - Business rules: BR-12, BR-55, BR-13, BR-14, BR-17, BR-55
  - Functionalities:
    - Select “Xóa” to trigger a confirmation dialog
    - On confirmation, send a DELETE request to the backend
    - Sends delete requests to the backend via API (axios.DELETE).
    - Refresh the list upon successful deletion
    - Display appropriate success or error messages

3.2.11 Workshop Management

3.2.11.1 Create Workshop

- **Function trigger:** This function is triggered when the **Manager** clicks the **“Quản lý Workshop”** button in the sidebar, and then selects **“Tạo Workshop”** to begin the multi-step workshop creation process.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows a Manager to create and configure a new pizza workshop event, including setting schedule, registration time, participation capacity, and pizza menu.
  - **Interface:**

- **Thông tin cơ bản** (Basic Info): name, header, description, location, organizer, hotline, zone
- **Thời gian** (Time): event date, registration open/close
- **Thông tin đăng ký** (Registration Info): fee, participant limits, pizza limits
- **Thực đơn** (Menu): select one or more pizza dishes

- **Data processing**:
  - **Input Validation**: Ensures the selected absence ID exists and is valid.
  - **API Call**: Sends a request to delete staff absences from the backend
  - **Feedback**: Displays success or error messages based on the API response.

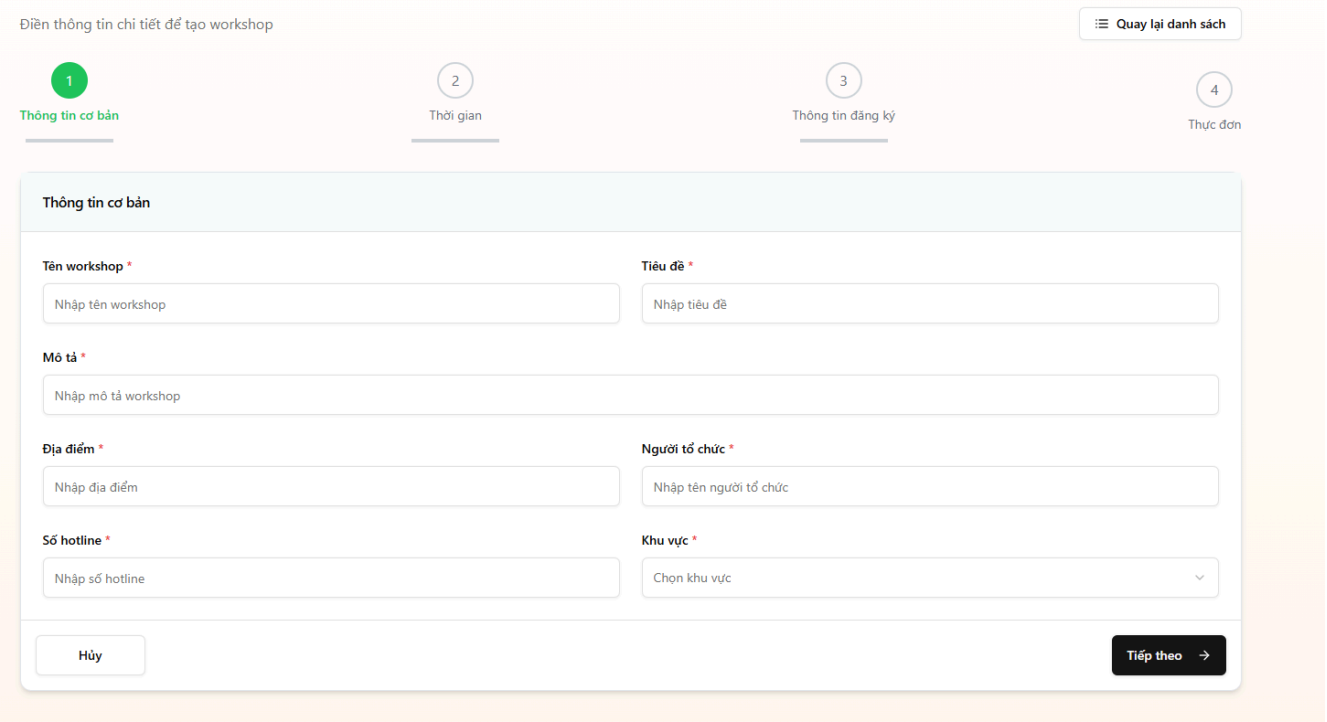


Figure 54.1- Create Workshop, Phase 1

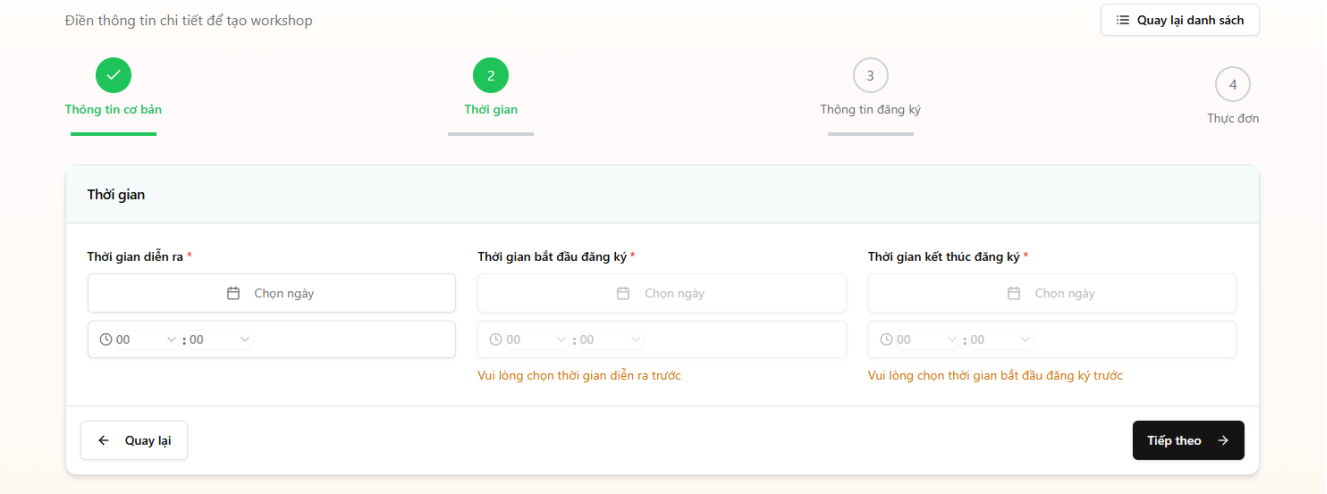


Figure 54.2- Create Workshop, Phase 2

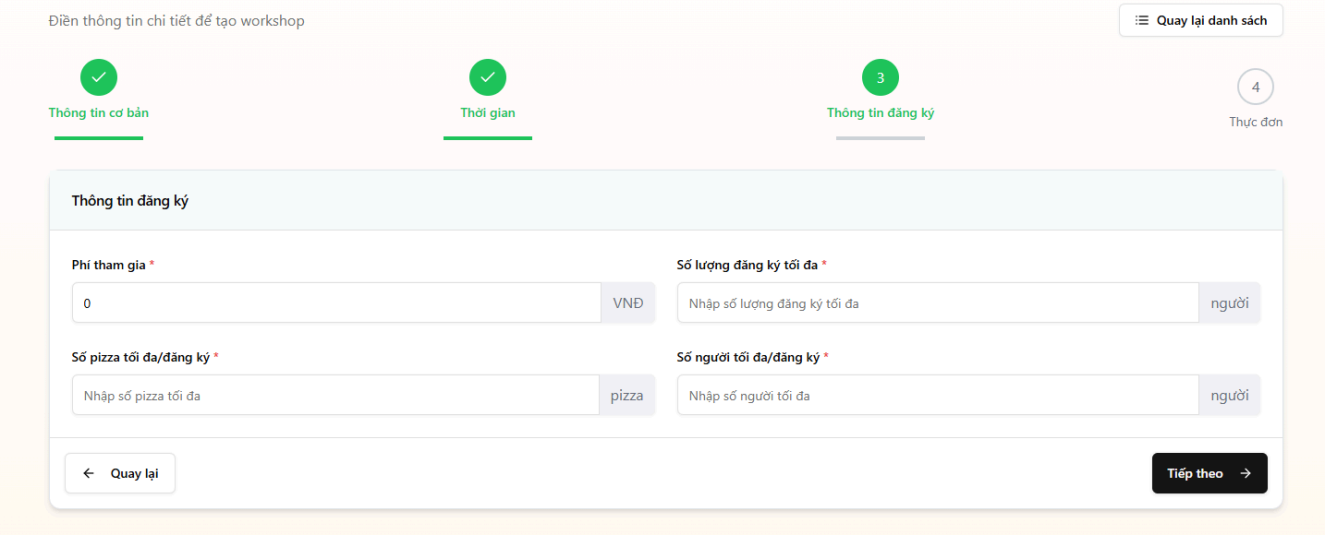


Figure 54.3- Create Workshop, Phase 3

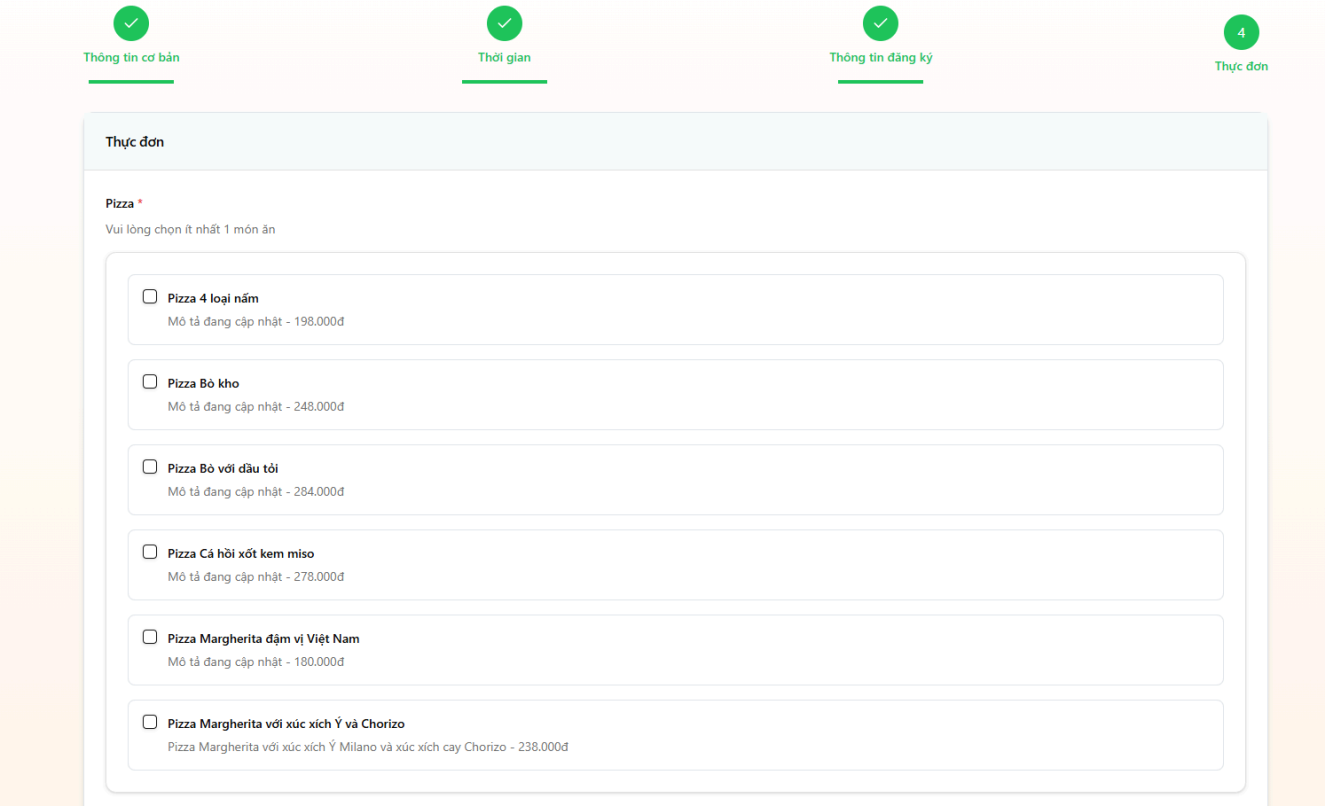


Figure 54.4- Create Workshop, Phase 4

- **Function Details:**
  - Data:
    - Absence ID (GUID): Unique identifier of the absence record to be deleted
  - Validation: Ensures the selected absence ID exists and is valid.
  - Business rules: BR-12, BR-47, BR-54
  - Functionalities:
    - Select “Xóa” to trigger a confirmation dialog

- On confirmation, send a DELETE request to the backend
- Sends delete requests to the backend via API (axios.DELETE).
- Refresh the list upon successful deletion
- Display appropriate success or error messages

3.2.11.1 View Workshop

- **Function trigger:** This function is triggered when the Manager logs into their account and clicks the “Quản lý Workshop” button in the sidebar. The system then loads the list of all created workshops.

- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows Managers to view and manage all existing workshop events, including their status, schedule, registration period, location, and participant capacity. From this screen, Managers can navigate to **edit**, **view detail**, or **cancel** functions.
  - **Interface:**
    - **Header**
    - **WorkshopDate**
    - Registration Period**(StartRegisterDate/ EndRegisterDate)**
    - **Location & Zone**
    - **TotalFee**
    - **Capacity status (e.g., 2/3 registered)**
    - **Status Badge**: (Scheduled, OpeningToRegister, ClosedRegister, Opening, Closed, Cancelled)
  - **Data processing**:
    - **Input Validation**: Ensures the selected absence ID exists and is valid.
    - **API Call**: Sends a request to delete staff absences from the backend
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

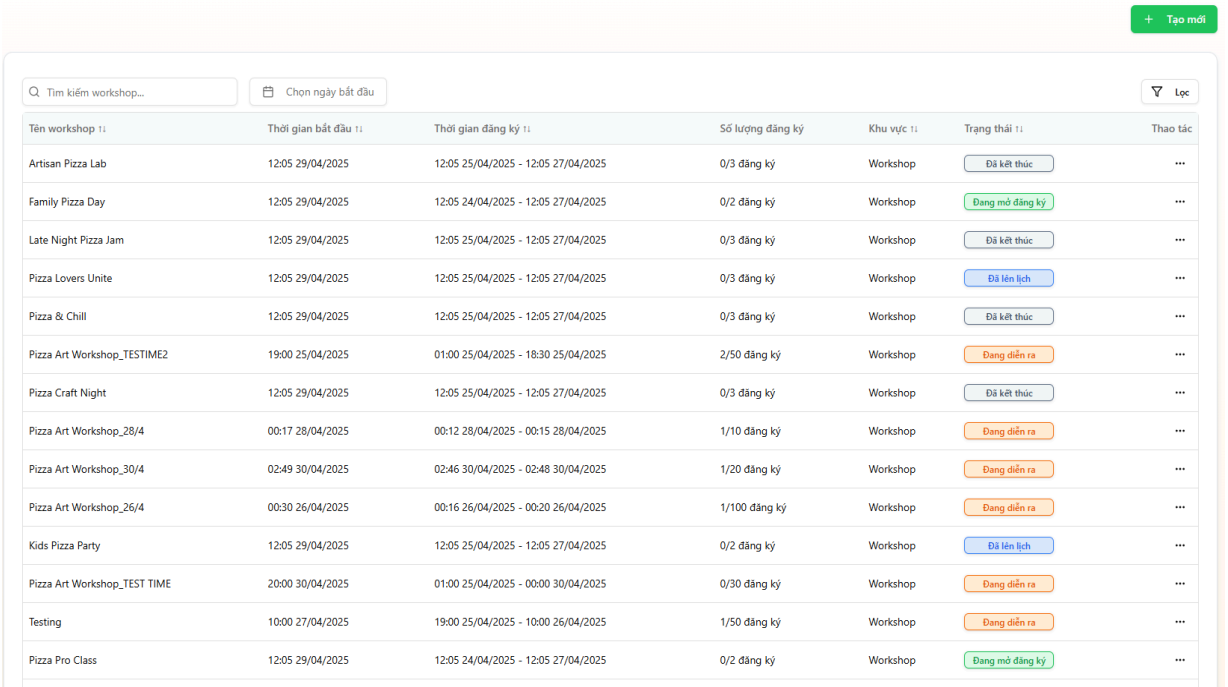


Figure 55- View Workshop

- **Function Details:**

- Data:
  - Absence ID (GUID): Unique identifier of the absence record to be deleted
- Validation: Ensures the selected absence ID exists and is valid.
- Business rules: BR-12, BR-47, BR-51
- Functionalities:
  - Select “Workshop” to redirect to a view page

3.2.11 Workshop Register Management

3.2.11.1 Register workshop

- **Function trigger:** This function is triggered when a customer visits the **Workshop Management** interface and clicks the “**Đăng Ký Ngay**” (Register Now) button for a workshop currently available during the registration period.
- **Function description:**
  - **Roles:** Customer
  - **Purpose:** Allows users to register for a workshop within a defined registration period, subject to capacity limits and validation checks.
  - **Interface:**
    - **Workshop Header** and **Description**
    - Ngày Diễn Ra (**WorkshopDate**)
    - Sức Chứa (**Capacity**): Limit per workshop
    - Thời Gian Đăng Ký (**StartRegisterDate**)
    - Phí Tham Gia (**TotalFee**)
    - Địa Điểm (**Location**)
    - Liên Hệ (**HotLineContact**)
  - **Data processing**:
    - **Input Validation**: Ensures the selected absence ID exists and is valid.
    - **API Call**: Sends a request to delete staff absences from the backend
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

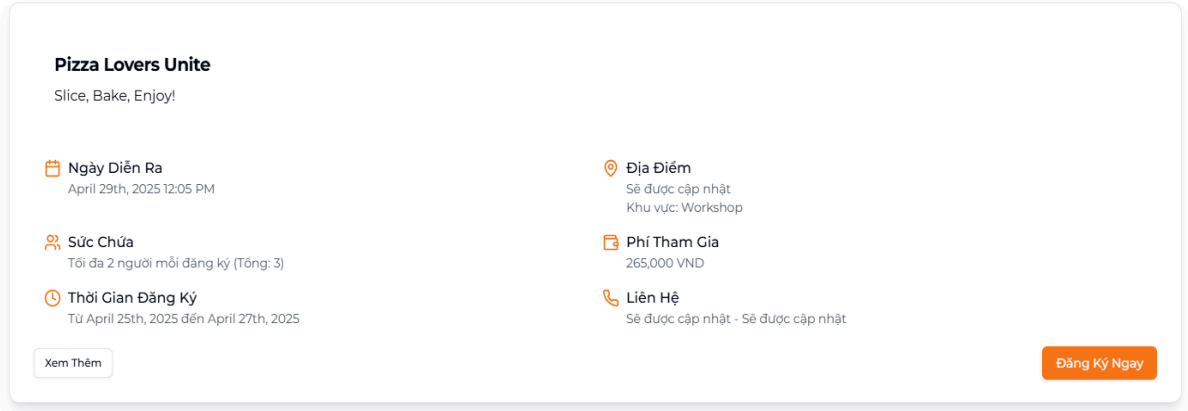


Figure 56- Register workshop

- **Function Details:**
  - Data:
    - Absence ID (GUID): Unique identifier of the absence record to be deleted
  - Validation: Ensures the selected absence ID exists and is valid.
  - Business rules: BR-12, BR-25, BR-26, BR-47
  - Functionalities:

        - Select “Đăng Ký Ngay” to trigger a check user information box
        - On confirmation on Phone Number, redirecting to a workshop details page

3.2.11.2 Check-in workshop

- **Function trigger:** This function is triggered when a customer arrives at the venue on the workshop day, and the Staff or Manager clicks the “Check-in” button for that customer’s registration.
- **Function description:**
    - **Roles:** Staff
    - **Purpose:** Allows authorized personnel to verify and check in registered participants for a workshop on the scheduled date. Successful check-in updates the participant’s status and confirms attendance for participation tracking.
    - **Interface:**
        - **List of Registered Participants for the Workshop**
        - **Workshop Details (read-only)**
        - **“Check-in” Button per Participant**
    - **Data processing**:
        - **Input Validation:** Ensures the participant is registered, not already checked in, and the current date matches the workshop date.
        - **API Call:** Sends a check-in request to the backend to update the participant’s attendance status.
        - **Feedback:** Displays success confirmation or error message based on result (e.g., already checked in, invalid registration).
- **Function Details:**
    - Data:
        - **WorkshopRegistrationId** (GUID, required): ID of the participant’s registration record.
        - CheckInStatus (enum): Updated to **“CheckedIn”**.
    - Validation:
        - **The registration must exist and be in Registered status.**
        - **Check-in must occur on the WorkshopDate.**
        - **Participants must not have been checked in previously.**
    - Business rules:
        - Only Staff or Managers may perform workshop check-ins.
        - Check-in is only allowed on the scheduled WorkshopDate.
        - A participant may only be checked in once per workshop.
        - The system should log check-in time and the user performing the check-in.
    - Functionalities:
        - Displays all registered participants for today's workshop.
        - Updates the registration record in real time (axios.PUT or axios.POST).
        - Shows check-in badge

3.2.12 Reservation Management

3.2.12.1 Create Reservation

- **Function trigger:** This function is triggered when a customer visits the **Reservation Management** interface and clicks the “**Đặt bàn ngay**” (Reserve now) button
- **Function description:**
    - **Roles:** Customer

- **Purpose:** Allows users to create a table reservation request by providing personal and booking details
- **Interface:**
  - **Reservation Header** and **Description**
  - Họ và tên (**Full name**)
  - Số điện thoại (**Phone number**)
  - Thời Gian đặt bàn (**Reservation Datetime**)
  - Số người (**Number of Customer**)
- **Data processing**:
  - **Input Validation**: Validates optional inputs (e.g., valid date format for filters).
  - **API Call**: Sends a request to reserve from the backend
  - **Feedback**: Displays success or error messages based on the API response.

- **Screen layout:**

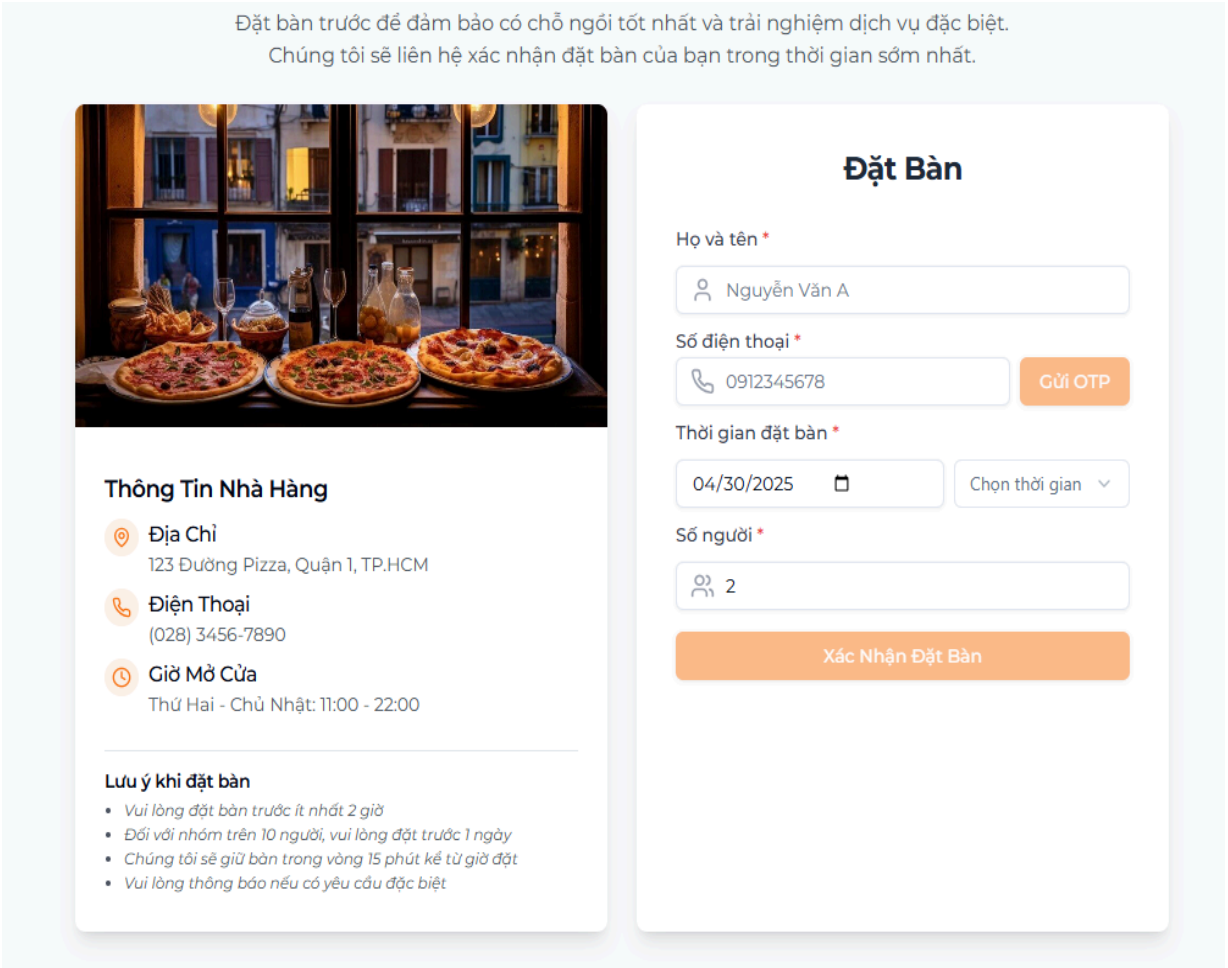


Figure 57-Create Reservation

- **Function Details:**
  - Data:
    - **CustomerName** (string, required): Full name of the customer making the reservation.
    - **PhoneNumber** (string, required): Phone number used for identification and OTP.
    - **BookingTime** (datetime, required): Date and time the customer wishes to reserve.
    - **NumberOfPeople** (int, required): Number of guests included in the reservation.

        - **ReservationPriorityStatus** (enum, system-generated): “Priority” or “NonPriority” based on slot capacity.
        - **ReservationStatusEnum** (enum, system-assigned): Default is “Confirmed”.
        - **CustomerId** (Guid?, optional): Linked if the customer already exists by phone number.
        - **TableId** (Guid?, optional): Null on creation, to be assigned later.
        - **AssignTableJobId** (string?, system-generated): Background job ID for auto table assignment.
    - Validation:
        - All required fields must be filled in before submission.
        - Phone number must be 10–11 digits and OTP-verified.
        - BookingTime must fall within a valid reservation slot configured in the system.
        - NumberOfPeople must be a positive integer (≥ 1).
        - The system checks if the selected slot is within capacity:
            - If yes → Priority
            - If no → NonPriority
        - Customers who don’t exist in the database are auto-created using the phone number.
    - Business rules:
        - BR-12: Access and role control must match the current user (Customer or Manager).
        - BR-31: The booking time must fall within a valid configured reservation slot.
        - BR-32: Each slot must not exceed configured capacity; priority determined accordingly.
        - BR-33: Auto-create customer if they don’t already exist.
        - BR-34: OTP verification required for customer phone number.
        - BR-35: Customers must arrive before the scheduled booking time to be checked in.
    - Functionalities:
        - Validates all fields before submission.
        - Determines ReservationPriorityStatus based on live slot capacity.
        - Triggers a real-time notification for new reservations using _realTimeNotifier.
        - Stores data into the backend via axios.POST.
        - On success, it shows a confirmation message and optionally starts the table assignment process.
        - On failure, displays contextual errors and prevents submission.
        - Updates reservation list (Manager) or shows confirmation UI (Customer).

3.2.12.2 Confirm Reservation

- **Function trigger:** This function is triggered when a manager clicks the “Xác nhận đặt bàn” (Confirm Reservation) button from the pending reservation list.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows users to confirm a pending reservation by updating its status to “Confirmed” after validation.
    - **Interface:**
        - **Reservation Information**
        - **Customer Details**
        - **“Confirm Reservation” Button**
    - **Data processing**:

        - **Input Validation**: Validates the existence and current state of the reservation.
        - **API Call**: Sends a request to update the reservation status from “Created” to “Confirmed”.
        - **Feedback**: Displays success or error messages based on the API response.
- **Function Details:**
    - Data:
        - **ReservationId (Guid, required): Unique identifier of the reservation to be confirmed**
    - Validation:
        - The ReservationId must exist and be in a “Created”
        - The current user must be authorized to confirm the reservation (Customers can only confirm their own).
        - The reservation’s BookingTime must still be in the future.
    - Business rules:
        - Only reservations in a “Created”
        - Manager can only confirm reservations.
        - The system must log the confirmation timestamp and user for audit purposes.
        - Reservations with a past BookingTime cannot be confirmed.
        - The status can only be updated if all validation rules are satisfied.
    - Functionalities:
        - Validates the reservation and current status.
        - Updates ReservationStatusEnum to “Confirmed”.
        - Records ConfirmedAt and ConfirmedByUserId.
        - Sends the update to the backend using axios.PUT.
        - Displays appropriate success or error messages.
        - Updates the reservation list (for Managers) or shows confirmation UI (for Customers).
        - Triggers real-time notifications for confirmed reservations (if applicable).

3.2.12.3 Assign Table Reservation

- **Function trigger:** This function is triggered when a Manager manually assigns a table to a previously confirmed reservation, typically around 45 minutes before the customer’s arrival time.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows a Manager to assign a specific **closed** table to a **confirmed** reservation. Upon assignment, the table’s status is updated to **“Reserved”**, preparing it for the customer’s arrival and enabling order placement when the customer checks in.
    - **Interface:**
        - **Reservation details (read-only)**
        - **List of closed tables eligible for assignment**
        - **“Assign Table” button**
    - **Data processing**:
        - **Input Validation:** Verifies that the selected table is closed, not already assigned, and has enough capacity.
        - **API Call:** Assigns the table to the reservation and updates the table status to “Open”.
        - **Feedback:** Displays a success message or error notification if validation fails.
- **Function Details:**
    - Data:

        - **ReservationId** (Guid, required): The confirmed reservation to be assigned a table.
        - **TableId** (Guid, required): The selected closed table.
        - **TableStatusEnum** (enum, system-updated): Updated from "Closed" to "Reserved".
    - Validation:
        - Reservation must be in "Confirmed" status.
        - Current time must be within 45 minutes of the reservation's BookingTime.
        - Selected table must be in "Closed" status and unassigned.
        - Table capacity must accommodate the number of guests.
        - Table must not already be reserved or open.
    - Business rules:
        - Only Managers can assign tables to confirmed reservations
        - Table assignment should occur around 45 minutes prior to reservation time
        - Once assigned, table status is updated to **"Reserved"**
    - Functionalities:
        - Lists confirmed reservations with BookingTime within reservation slot
        - Displays closed tables eligible for assignment.
        - Assigns selected table and updates its status to "Reserved".
        - Saves assignment data including timestamp and manager ID.
        - Sends updates via axios.PUT or axios.POST.
        - Triggers optional UI update or real-time notification.

3.2.12.4 Check-in Table Reservation

- **Function trigger:** This function is triggered when a customer arrives at the restaurant and the Manager clicks the **"Check-in"** button for their assigned and reserved table reservation.
- **Function description:**
    - **Roles:** Manager, staff
    - **Purpose:** Allows the Manager to check in a customer with a valid reservation. Upon check-in, the reservation is marked as **"CheckedIn"**, and the assigned table's status is updated from **"Reserved"** to **"Opening"** to enable order placement.
    - **Interface:**
        - **List of today's reservations**
        - **Reservation and customer information (read-only)**
        - **"Check-in" button**
    - **Data processing**:
        - **Input Validation:** Ensures the reservation is valid, in "Confirmed" status, and assigned to a table with "Reserved" status.
        - **API Call:** Updates reservation status to "CheckedIn" and table status to "Opening".
        - **Feedback:** Displays a success confirmation or error if check-in conditions aren't met.
- **Function Details:**
    - Data:
        - **ReservationId (Guid, required):** ID of the reservation being checked in.
        - **TableId (Guid, required):** ID of the reserved table.
        - **ReservationStatusEnum (enum, system-updated):** Updated to "CheckedIn".
        - **TableStatusEnum (enum, system-updated):** Updated from "Reserved" to "Opening".
    - Validation:

        - Reservation must be in **“Confirmed”** status
        - Reservation must already be assigned to a table.
        - Assigned table must be in **“Reserved”** status.
    - Business rules:
        - Only Managers and Staff can perform check-in for reservations.
        - A reservation must be assigned to a table before it can be checked in.
        - Upon check-in, table status must change from “Reserved” to “Opening”.
        - A reservation may only be checked in once.
    - Functionalities:
        - Displays a filtered list of today's confirmed + reserved reservations.
        - Allows check-in with one click, updating both reservation and table states.
        - Sends data via axios.PUT to the backend.
        - Shows confirmation message and optionally highlights the table as in-use.
        - Enables the ordering interface for the newly opened table.

3.2.13 Voucher Batch Management

3.2.13.1 Create Voucher Batch

- **Function trigger:** This function is triggered when a Manager logs into their account, navigates to the "**Khuyến mãi**" module, clicks the "**Loại Voucher**" tab, and then selects the "**Thêm loại voucher**" button.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows the manager to create and register a new voucher batch with specified details including batch code, discount type, description, validity period, quantity, and discount value.
    - **Interface:** Form View for adding a new voucher batch
    - **Data processing**:
        - **Input Validation**:
            - **Voucher Batch Code**: Must be a unique, non-empty string.
            - **Discount Type**: Must be selected from available options (either “Phần trăm (%)” or “Trực tiếp (VNĐ)”).
            - **Start Date**: Must be a valid date, not in the past, and must be ≤ End Date.
            - **End Date**: Must be a valid date, not in the past, and must be ≥ Start Date.
            - **Quantity**: Must be a positive integer.
            - **Discount** Value:
                - The percentage must be between 1–100.
                - If a fixed amount (VNĐ), must be a positive number.
        - **API Call**: Sends a request to create voucher batch from the backend
        - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

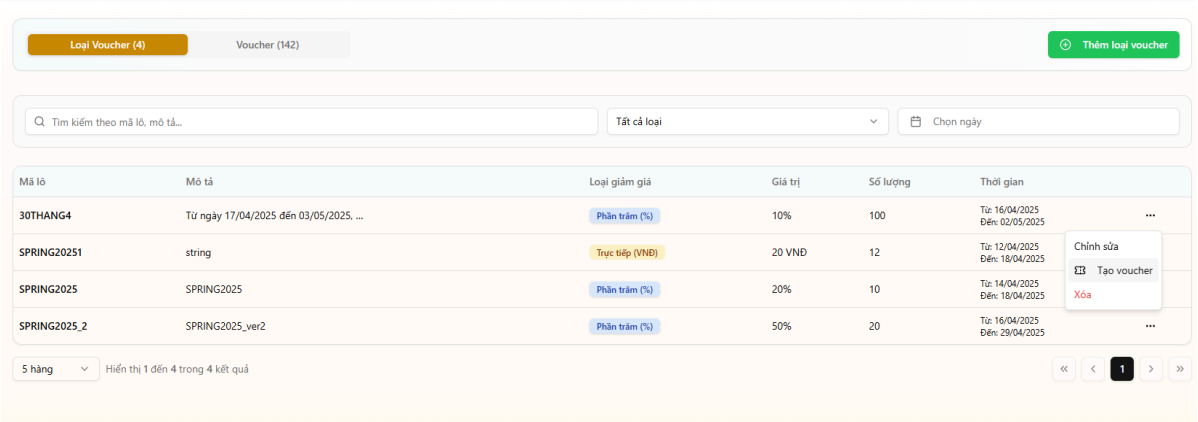


Figure 58- Create Voucher Batch

- **Function Details:**
  - Data:
    - Code (string, required): Unique code identifying the voucher batch.
    - Description (string, optional): Text describing the promotion.
    - Discount Type (enum, required): “Phần trăm (%)” or “Trực tiếp (VNĐ)”.
    - Discount Value (decimal, required): Value of the discount.
    - Quantity (int, required): Number of vouchers in the batch.
    - Start Date (datetime, required): Validity start date.
    - End Date (datetime, required): Validity end date.
  - Validation:
    - All required fields (except description) must be filled in before submitting.
    - The date range must be valid: Start Date must be less than or equal to End Date.
    - Quantity must be a positive integer.
    - Discount value must be:
    - A percentage between 1–100 if the discount type is percentage.
    - A positive monetary value if the discount type is a fixed amount.
    - Batch code must be unique (if editable) or must already exist in the system.
    - Data must be submitted only if the user is authenticated and authorized to perform the action.
  - Business rules:
  - Functionalities:
    - After successful update, the updated batch appears in the voucher list.
    - Allows the manager to fill in all necessary fields and submit.
    - On “Cancel”, aborts the operation and returns to the voucher list
    - On success, updates the list to show the newly added batch.

3.2.13.2 View voucher batch

- **Function trigger:** This function is triggered when a Manager logs into their account, navigates to the "**Khuyến mãi**" module, and clicks on the "**Loại Voucher**" tab.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** View and manage all existing voucher batches with filtering and search capabilities.
  - **Interface:** List of voucher batch
  - **Data processing**:

- **Input Validation**:
- **API Call**: Sends a request to get voucher batch from the backend
- **Feedback**: Displays success or error messages based on the API response.

- **Screen layout:**

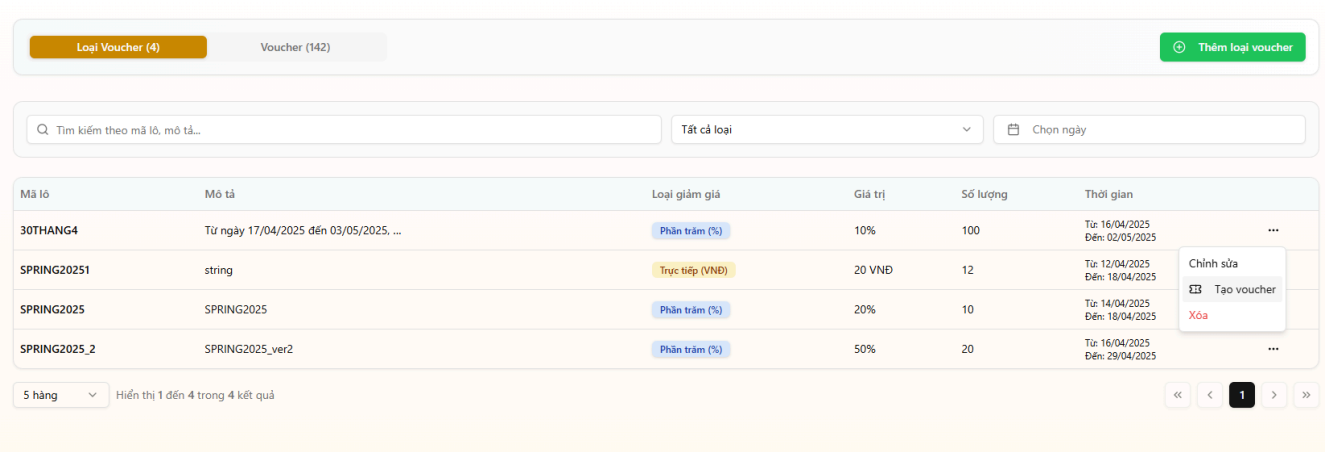


Figure 59- View voucher batch

- **Function Details:**
  - Data:
    - VoucherBatch ID (GUID): Unique identifier of the voucher batch.
    - Code (string): Code associated with the voucher batch.
    - Description (string): Description of the promotion.
    - Discount Type (enum): Either percentage (%) or fixed amount (VNĐ).
    - Discount Value (decimal): Value of the discount.
    - Quantity (int): Total number of vouchers in the batch.
    - Start Date (datetime): Start date of the voucher batch.
    - End Date (datetime): End date of the voucher batch.
  - Validation:
    - Data must be fetched from the backend.
      Date range filters, if used, must be valid and in logical order (start date ≤ end date).
  - Business rules:
  - Functionalities:
    - Fetch and display all voucher batches in a paginated list.
    - Supports searching by batch code or description
    - Supports filtering by discount type and date range
    - Displays key info per batch: code, description, discount type/value, quantity, usage dates.

3.2.13.3 Edit Voucher Batch

- **Function trigger:** This function is triggered when a Manager logs into their account, navigates to the "**Khuyến mãi**" module, and clicks on the "**Loại Voucher**" tab. When the Manager clicks the "**...**" button on a voucher batch in the list and selects "**Chỉnh sửa**".
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows the manager to update the details of an existing voucher batch, such as code, type, description, quantity, discount value, and validity period.
  - **Interface:** Form view of update voucher batch

- **Data processing**:
  - **Input Validation**:
    - **Voucher Batch Code**: The voucher batch code must be a unique string and cannot be empty
    - **Discount Type**: The discount type must be selected from the available options (either "Phần trăm (%)" or "Trực tiếp (VNĐ)")
    - **Start Date**: The start date must be a valid date and cannot be in the past. It must be earlier than or equal to the end date
    - **End Date**: The end date must be a valid date and cannot be in the past. It must be later than or equal to the start date
    - **Quantity**: The quantity must be a positive integer
    - **Discount** Value: The discount value must be a positive integer
  - **API Call**: Sends a request to update voucher batch from the backend
  - **Feedback**: Displays success or error messages based on the API response.

- **Screen layout:**

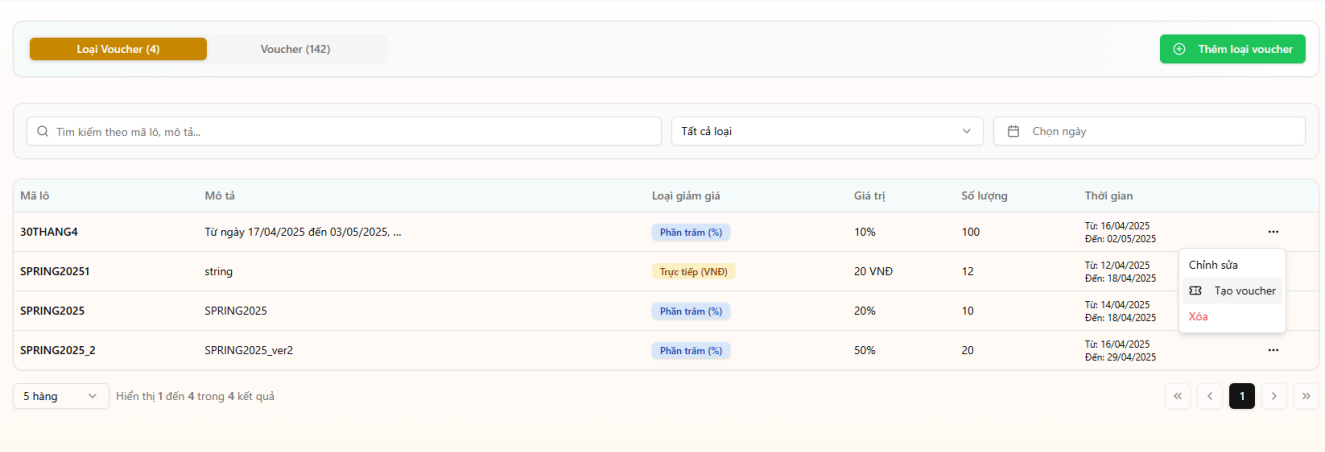


Figure 60- Edit Voucher Batch

- **Function Details:**
  - Data:
    - VoucherBatch ID (GUID): Unique identifier of the voucher batch.
    - Code (string): Code associated with the voucher batch.
    - Description (string): Description of the promotion.
    - Discount Type (enum): Either percentage (%) or fixed amount (VNĐ).
    - Discount Value (decimal): Value of the discount.
    - Quantity (int): Total number of vouchers in the batch.
    - Start Date (datetime): Validity start date of the voucher batch.
    - End Date (datetime): Validity end date of the voucher batch.
  - Validation:
    - All required fields (except description) must be filled in before submitting.
    - The date range must be valid: Start Date must be less than or equal to End Date.
    - Quantity must be a positive integer.
    - Discount value must be:
    - A percentage between 1–100 if the discount type is percentage.
    - A positive monetary value if the discount type is a fixed amount.
    - Batch code must be unique (if editable) or must already exist in the system.

        - Data must be submitted only if the user is authenticated and authorized to perform the action.
    - Business rules:
    - Functionalities:
        - Users can update key fields: Batch Code, Discount Type, Description, Start Date, End Date, Quantity, and Discount Value.
        - On "Cancel", the edit is aborted, and the user is returned to the voucher list view.
        - After successful update, the updated batch appears in the voucher list.

3.2.13.4 View voucher

- **Function trigger:** This function is triggered when a Manager logs into their account, navigates to the "**Khuyến mãi**" module, and selects the "**Voucher**" tab.
- **Function description:**
    - **Roles:** Manager
    - **Purpose:** Allows the manager to view a list of all vouchers generated from voucher batches. The list supports filtering, searching by code, pagination, and viewing individual voucher status.
    - **Interface:** Tabular view displaying voucher data with filtering and search tools
    - **Data processing**:
        - **Input Validation**:
        - **API Call**: Sends a request to get voucher from the backend
        - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

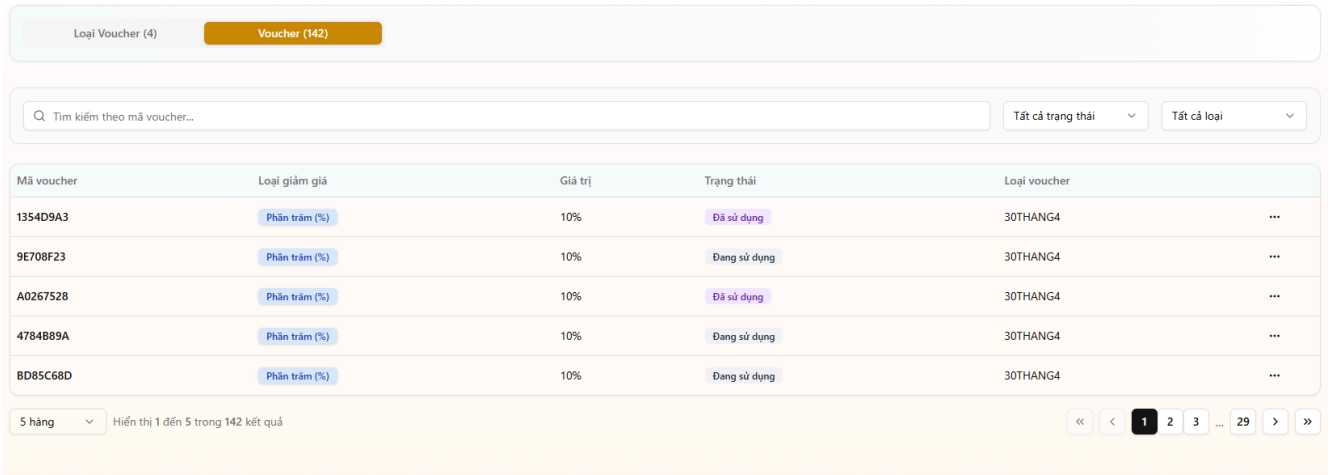


Figure 61- View voucher

- **Function Details:**
    - Data:
        - Voucher ID (string): Unique identifier of the individual voucher.
        - Code (string): Code assigned to each voucher.
        - Discount Type (enum): Either percentage (%) or fixed amount (VNĐ).
        - Discount Value (decimal): The value of the discount (e.g., 10% or 50,000 VNĐ).
        - Status (enum): Usage status of the voucher (e.g., Not Used, In Use, Used).
        - Voucher Batch Code (string): The batch code from which the voucher was generated.

- Validation:
  - Data must be fetched from the backend with valid authentication.
  - If search or filter is applied:
    - Search text must be trimmed and matched against valid voucher codes.
    - Status and batch filters must match predefined valid values.
- Business rules:
- Functionalities:
  - Fetch and display all voucher batches in a paginated list.
  - Pagination with controls to navigate between pages.
  - Action menu per voucher for potential operations.
  - Display total number of results and current page range.

3.2.13.5 Delete voucher

- **Function trigger:** This function is triggered when a Manager navigates to the "Voucher" tab in the "**Khuyến mãi**" module, clicks the "**...**" button next to a voucher in the list, and selects "**Xóa**" from the dropdown menu.
- **Function description:**
  - **Roles:** Manager
  - **Purpose:** Allows authorized users to permanently delete a specific voucher from the system if it has not been used.
  - **Interface:** Confirmation Dialog for deletion.
  - **Data processing**:
    - **Input Validation**:
      - The Voucher ID must be a valid GUID (Globally Unique Identifier).
      - The system must ensure that the voucher exists in the database before proceeding with the deletion.
    - **API Call**: Sends a request to delete voucher from the backend
    - **Feedback**: Displays success or error messages based on the API response.
- **Screen layout:**

Figure 62- Delete voucher

- **Function Details:**
  - Data:
    - Voucher ID (GUID): Unique identifier of the voucher to be deleted.
  - Validation:
    - The backend API must verify that the voucher with the provided Voucher ID exists in the system.
  - Business rules:
  - Functionalities:
    - Confirmation Dialog: Confirms the deletion.
    - API Call: Send delete request to backend.
    - Backend Validation: Ensures voucher is eligible for deletion.

- Feedback & UI Update: Displays success or error message, and updates list.
- Pagination: Updates pagination if necessary after deletion.

3.2.14 Feedback Management

3.2.14.1 Create Feedback

- **Function trigger:** This function is triggered when the customer completes an order and opens the post-dining feedback modal.
- **Function description:**
  - **Roles:** Customer
  - **Purpose:** To collect and store customer feedback after an order is completed, including rating, optional tags for dissatisfaction, and a comment.
  - **Interface:**
    - **Star rating (1 to 5 stars)**
    - **Optional dissatisfaction tags (e.g., "Vệ sinh không sạch sẽ", "Giá không phù hợp với chất lượng", etc.)**
    - **Comment box**
    - **Optional phone number for callback**
    - **Button: "Gửi đánh giá"**
  - **Data processing**:
    - **Input Validation**:
      - The Voucher ID must be a valid GUID (Globally Unique Identifier).
      - The system must ensure that the voucher exists in the database before proceeding with the deletion.
    - **API Call**: Sends a request to delete voucher from the backend
    - **Feedback**: Displays success or error messages based on the API response.

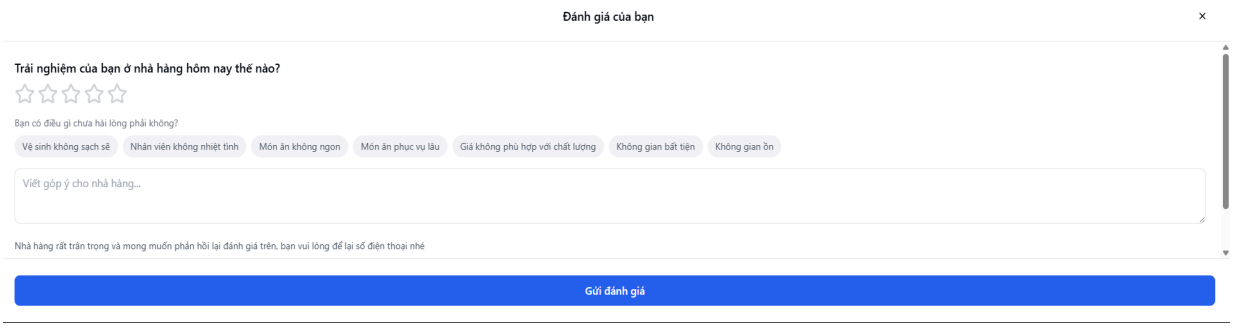


Figure 63- Create Feedback

- **Function Details:**
  - Data:
    - **Rating** (int, required): A score between 1 and 5 representing the customer's overall satisfaction.
    - **Comments** (string, optional): Free-text area for feedback content.
    - **FeedbackDate** (datetime, system-generated): Timestamp when the feedback was submitted.
    - **OrderId** (Guid, required): ID of the related order for which feedback is being given.
  - Validation:
    - Rating must be between 1 and 5.
    - Comments are optional but trimmed for whitespace.

        - Order ID must exist and be valid.
        - Only one feedback record per order is allowed.
        - Users must be authenticated to submit feedback.
    - Business rules:
        - **BR-12:** Users can only submit feedback if their role allows it (Customer/Manager).
        - **BR-51:** Feedback entries are tracked for historical records and reporting.
    - Functionalities:
        - Displays a rating interface and optional comment field.
        - Validates user input before submission.
        - Prevents duplicate feedback for the same order.
        - Sends valid data to the backend using axios.POST.
        - Records the feedback date using server time.
        - Displays a success or error message depending on API response.
        - Optionally notifies the Manager for follow-up if low rating and phone number is provided.
        - Closes the modal or refreshes the page upon success.

3.2.15 Notification Management

3.2.15.1 Create Notification

- **Function trigger:** This function is triggered when a customer or system user clicks the “Gọi nhân viên” button and submit a request with a reason (e.g., “Lấy thêm bát đũa, dọn bàn…”).
- **Function description:**
    - **Roles:** Customer
    - **Purpose:** To notify zone-assigned staff in real time when assistance is needed at a specific table. The notification is both sent via SignalR and stored for historical viewing.
    - **Interface:**
        - Text Input: Reason for calling staff
        - Submit Button: “Gửi yêu cầu” (disabled until text is entered)
        - Modal title: “Gọi nhân viên”
        - Closes automatically after successful submission
    - **Data processing**:
        - **Input Validation**:
            - The Voucher ID must be a valid GUID (Globally Unique Identifier).
            - The system must ensure that the voucher exists in the database before proceeding with the deletion.
        - **API Call**: Sends a request to delete voucher from the backend
        - **Feedback**: Displays success or error messages based on the API response.

Figure 64- Create Notification

- **Function Details:**
    - Data:

        - **Table** (string, required): Code or name of the table where the notification originates (e.g., "B02").
        - **Zone** (string, required): Zone in which the table is located (e.g., "Zone_A").
        - **Note** (string, optional): Custom reason for requesting assistance (e.g., "Dọn bàn", "Lấy thêm nước").
        - **Type** (enum, required): Notification type (e.g., `CallStaff`).
        - **Title** (string, system-generated): "Gọi nhân viên".
        - **Message** (string, system-generated or custom): Descriptive message shown in the staff interface.
        - **Payload** (string): Identifier (zone name) used to route real-time notifications.
        - **RequiresPersistence** (bool): Indicates that the notification should be stored and shown in the bell icon.
        - **CreatedAt** (datetime, system-generated): Time of creation.
        - **IsHandled** (bool): Initially `false`, updated when the notification is processed.
    - Validation:
        - Table and zone must be valid and currently active.
        - Optional notes must not exceed character limits (if any).
        - The sender must be authenticated (customer or system).
        - A valid zone must be assigned for routing.
        - Duplicate unhandled notifications should not be created for the same table/zone in rapid succession (rate-limiting recommended).
    - Business rules:
        - **BR-12:** Only authenticated users with valid roles can perform actions.
        - **BR-52:** Notifications are routed to staff members assigned to the target zone.
    - Functionalities:
        - Opens a modal for customers to input their assistance request.
        - Validates input and ensures a zone and table are identified.
        - Sends the request to the backend via axios.POST.
        - Backend calls SendStaffCallNotificationAsync() in NotificationService.
        - Notification is sent via SignalR and saved for staff to view.
        - A bell icon indicator or dashboard displays active notifications for zone staff.
        - Provides immediate UI feedback on successful or failed submission.

3.2.16 Authentication Management

3.2.16.1 Login

- **Function trigger:** This function is triggered when a user wants to login their account

- **Function description:**
  - Roles: Manager, Staff, Chef
  - Purpose: Enables users to access their accounts based on credentials.
  - Interface: Login form with fields for username and password
  - Data processing:
    - User Input
    - Validation
    - Session Handling

- **Screen layout:**

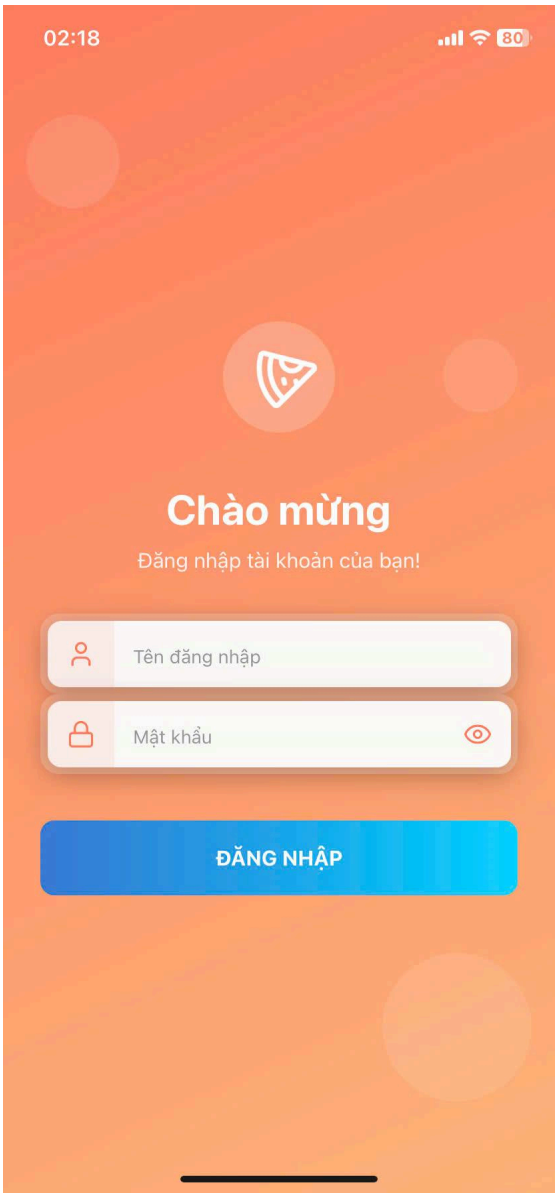


Figure 65- Login

- **Function Details:**
  - Data:

        - Username
        - Password
    - Validation: Ensure credentials match records in the database
    - Business rules:
        - **BR-01:** All users must be logged in to access the system.
        - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - Functionalities:
        - Credential Validation: Authenticate user input.
        - Session Management: Start or restore user session.
        - Error Handling
        - Security: Encrypt passwords during transmission.

3.2.16.2 Logout

- **Function trigger:** Triggered when a user clicks the "Đăng xuất" button.
- **Function description:**
    - Roles: Manager, Staff, Chef
    - Purpose: Ends the user session and ensures secure logout from the platform.
    - Interface: Logout button on the dashboard or navigation menu
    - Data processing:
        - Session Termination: Invalidate the user session.
        - Redirection: Redirect users to the homepage.
- **Screen layout:**

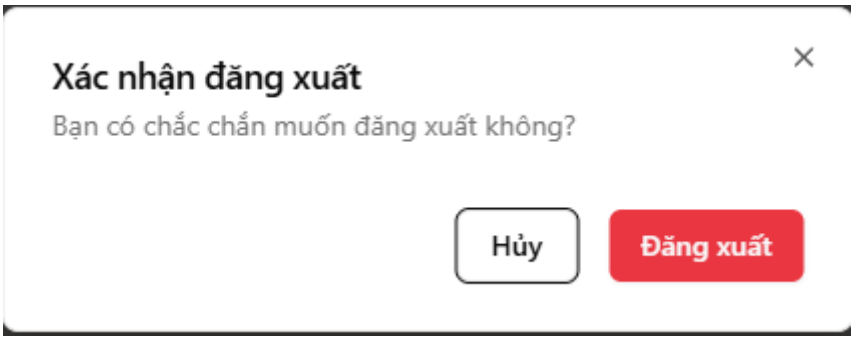


Figure 66- Logout

- **Function Details:**
    - Data:
        - User session data
    - Validation: Ensure that the session being terminated belongs to the logged-in user.
    - Business rules:
        - **BR-01:** All users must be logged in to access the system.
        - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - Functionalities:
        - Session Management: Securely terminate user sessions.
        - Redirection: Direct users to an appropriate page post-logout.

3.2.16.3 Change password

- **Function trigger:** Triggered when a user click the button "**Đổi mật khẩu**" from their account settings.

- **Function description:**
  - Roles: Manager, Staff, Chef
  - Purpose: Enables users to update their password securely.
  - Interface: Form with fields for old password, new password, and confirm password.
  - Data processing:
    - Old Password: Current password.
    - Password Update: Save the new password securely.
- **Screen layout:**

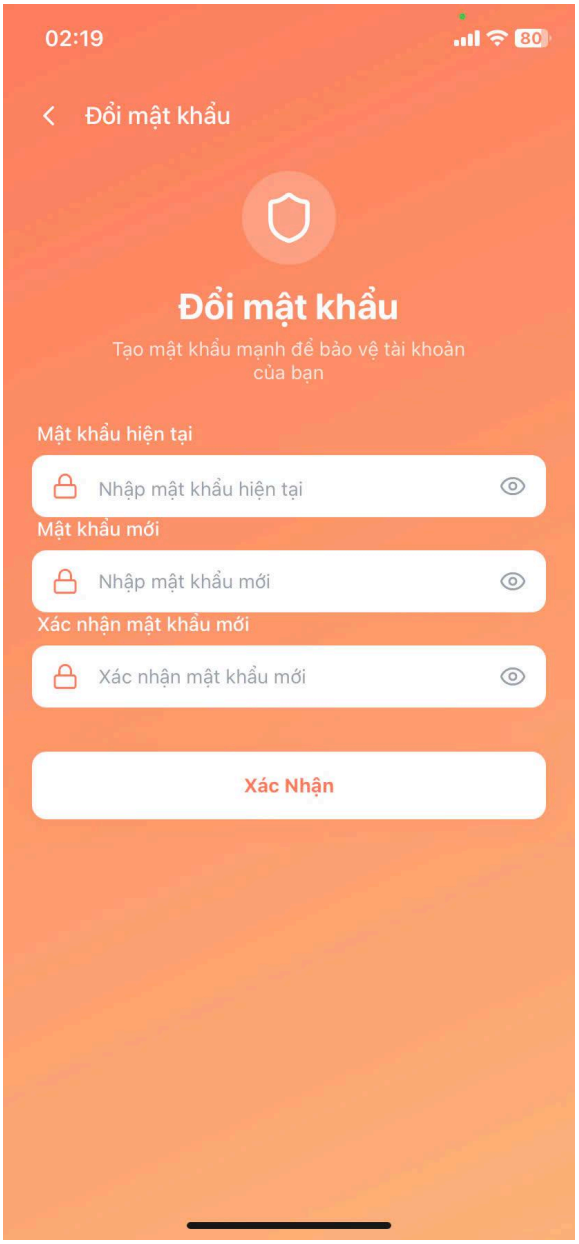


Figure 67- Change password

- **Function Details:**
  - Data:
    - Old and new passwords
  - Validation:
    - New password must meet complexity requirements.

        - Old password must match the current one.
    - Business rules:
        - **BR-01:** All users must be logged in to access the system.
        - **BR-11:** Manager can change their master password to ensure account security.
        - **BR-12:** Each user only has access rights and performs activities according to their assigned role.
    - Functionalities:
        - Old Password Validation: Authenticate user with their current password.
        - Password Update: Save the new password securely.

3.2.16.4 Reset account

- **Function trigger:** Triggered when Manager selects a staff member from the staff management screen and clicks the "**Chỉnh sửa**" button.
- **Function description:**
    - Roles: Manager
    - Purpose: Allows Manager to reset a staff member's account by updating personal information (name, phone, email) and setting a new password.
    - Interface: A form with editable fields for Full Name, Phone, Email, Password, and dropdowns for Staff Type and Status.
    - Data processing: Sends updated staff information including new password to the backend for storage.
- **Screen layout:**

Figure 68-Reset account

- **Function Details:**
    - Data:
        - **Id** (string, required): Unique identifier of the staff.
        - **FullName** (string, required): Updated full name of the staff.
        - **Phone** (string, required): Updated phone number.
        - **Email** (string, required): Updated email address.
        - **Password** (string, required): New password to be set.
        - **Staff Type** (int, required): Selected staff type (e.g., Manager, Staff).
        - **Status** (int, required): Selected staff status (e.g., full-time, part-time).
    - Validation:
        - **Password:**
            - Must be at least 8 characters if provided.
            - Must contain at least one uppercase letter if provided.
            - Must contain at least one special character if provided.
        - **Full Name:**
            - Required and must not be empty
            - Must have at least 2 characters
        - **Email:**
            - Required and must not be empty
            - Must follow a valid email pattern
        - **Phone**

        - Required.
        - Must contain only digits.
        - Must be at least 10 digits.

    - **Staff Type:**
        - Must be selected from a predefined list.
    - **Status:**
        - Must be selected from a predefined list.
- Business rules:
    - BR-01: All users must be logged in to access the system.
    - BR-03: Manager can create, edit, and delete Staff and Chef accounts.
    - BR-12: Each user only has access rights and performs activities according to their assigned role.
- Functionalities:
    - Old Password Validation: Authenticate user with their current password.
    - Password Update: Save the new password securely.

4. Non-Functional Requirements

4.1 External Interfaces

4.1.1 User Interfaces

- The user interface (UI) is designed to be responsive across multiple screens, ensuring that the main content is displayed without requiring horizontal scrolling.
- Web-Based Access: The system must be accessible via any standard web browser (e.g., Chrome, Firefox, Safari) without requiring plugins or extensions.
- Mobile Accessibility: A responsive design must be provided, making the system fully usable on mobile devices (Android and iOS).

4.1.2 Communications Interfaces

- The system can call an API through data transferred by HTTP.

4.2 Quality Attributes

4.2.1 Usability

- The web applications designed for chef and staff prioritize simplicity and ease of use, requiring minimal training while maintaining effective utilization.
- Managers and Managers should be able to navigate the web application with proficiency after receiving less than a day's worth of guided training..
- Critical tasks like login, data entry, and record searching should take under 5 seconds to complete on average.

4.2.2 Security

- Input data undergoes thorough validation before being stored in the database, ensuring data integrity.
- The system must use JSON Web Token with access and refresh token for secure access.
- Role-based access control (RBAC) must be implemented to manage user permissions.
- The system consistently performs authentication and authorization checks before executing any function, enhancing overall security.

4.2.3 Performance

- Average response time for key transactions (e.g., login, data entry) must be under 3 seconds.
- Maximum response time for complex operations (e.g., generating comprehensive financial reports) should not exceed 5 seconds.
- During peak operational periods, the system must efficiently process up to 100 transactions per second, ensuring responsive and reliable performance across large-scale restaurant operations.

5. Requirement Appendix

5.1 Business Rules

Code	Business Definition
BR-01	All users must be logged in to access the system.
BR-02	Manager can view a list of all account users in the system.
BR-03	Manager can create, edit, and delete Staff and Chef accounts.
BR-04	Manager can update information of menu and menu items, including information of the option for each menu item.
BR-05	Manager can update information of menu and menu items.
BR-06	Manager have the right to manage roles (Roles) in the system.
BR-07	Manager can assign or revoke permissions (Permissions) for each role.
BR-08	Only Managers can reactivate suspended or locked accounts
BR-09	Managers can view application activation reports.
BR-10	Manager has access to reports on sales revenue and other activities in the system.
BR-11	Manager can change their master password to ensure account security.

BR-12	Each user only has access rights and performs activities according to their assigned role.
BR-13	A staff member can only be scheduled for a shift if they have previously registered for that working slot on that specific day.
BR-14	A staff member can only be scheduled to one zone per working date and working slot
BR-15	Auto-assignment of zones can only begin on a Monday.
BR-16	Each full-time staff member is assigned to one zone (kitchen or dining) for the whole week.
BR-17	Each part-time staff member is assigned to a zone per day and per shift based on their approved registrations.
BR-18	Full-time chefs must be assigned to a kitchen area. If no kitchen zone is available, the system must reject the assignment.
BR-19	Full-time waiters or servers (non-chefs) must be assigned to a dining area. If no dining zone is available, the system must reject the assignment.
BR-20	A staff member can be assigned to a zone manually by defining the time range they will work in that zone.
BR-21	A shift swap can only occur between two existing staff members.
BR-22	Both staff members must have approved working slot registrations for the specified dates and shifts they wish to swap.
BR-23	The request is submitted at least 2 days before the earlier of the two shift dates being swapped.
BR-24	Staff has the ability to receive requests from customers while taking responsibility for that zone
BR-25	Registrations must be made within a 2 weeks range
BR-26	A staff member can only register for up to 3 shifts per week
BR-27	Each order code is a combination of the last 9 digits of ticks + 3 random digits.
BR-28	The voucher must exist and be claimed using point
BR-29	Only one voucher from the same batch can be applied to a single order
BR-30	Vouchers must have defined start and end validity dates.
BR-31	A booking can only be made if the booking time falls within an active reservation slot.
BR-32	Each reservation slot has limited capacity per day.
BR-33	A customer is auto-created if one with the provided phone number doesn't already exist.
BR-34	The customer must provide a valid OTP to confirm the booking.

BR-35	The customer must appear before booked time for checking-in
BR-36	A reservation must have a table assigned before checking in.
BR-37	Additional fee represents the reduction or addiction of the total fee before calculating other extra fee of service
BR-38	Reservation can be manually confirmed by staff.
BR-39	Menu items must have at least one size or variant before being published.
BR-40	Option items must be linked to a parent option and cannot exist standalone.
BR-41	Kitchen area will be divided into two main parts, which is hot section and pantry, that handle specific kind of food.
BR-42	A new reservation slot must not overlap with existing slots.
BR-43	Managers can disable vouchers before their expiration date if needed.
BR-44	The manager can add, view, update, and delete information about vouchers.
BR-45	Manager has the right to create, view, update and delete information about dishes and its options.
BR-46	Managers have the right to view statistics, sale reports, including order details, value and other related information.
BR-47	Workshop only include pizza and cost additional fee per workshop registration
BR-48	Manager has the right to create, view, update and delete information about pizzas and its options on the workshop.
BR-49	Managers and Staffs have the right to disable or enable table if needed, unless the table is occupied by customers
BR-50	The system will track voucher redemption history per customer to prevent fraudulent use.
BR-51	The system needs to record a history of all changes made by the Manager to information products, configurations and other data.
BR-52	The system will notify requests from customer to staff that handle the assigned zone or table
BR-53	When creating a staff account, the system must validate that all input fields meet format and security requirements (e.g., password strength, valid email/phone number, username uniqueness), and reject invalid input.
BR-54	Managers can create work shifts and working slots with valid time ranges and day-of-week assignments. Shift time must not overlap existing defined shifts for the same day, and part-time capacity must be a non-negative number.

BR-55	The system must prevent assigning a zone to any staff member marked as absent for a given working date and slot. Absence records must block zone assignment and be mutually exclusive with existing shift assignments.

Table 69- Business rules

5.2 Application Messages List

No	Message Code	Message Type	Context	Content
1	MSG01	In line	Additional Fee is invalid	Amount cannot be negative
2	MSG02	In line	Additional Fee not found	Additional Fee not found
3	MSG03	In line	Working Slot register status is invalid	Working Slot register status is invalid
4	MSG04	In line	Working Slot register not found	Working Slot register not found
5	MSG05	In line	Swap Working Slot request already rejected	Swap Working Slot request has already been rejected
6	MSG06	In line	Swap Working Slot request pending approval	Swap Working Slot request is pending manager approval
7	MSG07	In line	Swap Working Slot request already approved	Swap Working Slot request has already been approved
8	MSG08	In line	Swap Working Slot request not found	Swap Working Slot request not found
9	MSG09	In line	Swap Working Slot request invalid state	Cannot reject a Swap Working Slot request that is not pending
10	MSG10	In line	Config not found	Config not found
11	MSG11	In line	Config is invalid	Config is invalid
12	MSG12	In line	Role not found	Role not found
13	MSG13	In line	Working Slot already registered	Employee has already registered for this working slot
14	MSG14	In line	Shift not found	Shift not found
15	MSG15	In line	Current order ID exists	Table already has an existing order ID
16	MSG16	In line	Product Option not found	Product Option not found
17	MSG17	In line	Day of the week not found	Day of Week not found
18	MSG18	In line	Staff Schedule not found	Staff Schedule not found
19	MSG19	In line	Ingredient not found	Ingredient not found
20	MSG20	In line	Staff Zone Schedule not found	Staff Zone schedule not found

21	MSG21	In line	Working time not found	Working time not found
22	MSG22	In line	Table Booking not found	Table Booking not found
23	MSG23	In line	Pizza Size not found	Pizza Size not found
24	MSG24	In line	Booking not found	Reservation not found
25	MSG25	In line	Booking slot full	No available Tables at this time
26	MSG26	In line	Invalid booking status	Invalid Reservation Status
27	MSG27	In line	Invalid booking capacity	Guest Count exceeds Table Capacity
28	MSG28	In line	Table not yet assigned	Table not yet assigned
29	MSG29	In line	Booking slot not found	Booking is not available at this time
30	MSG30	In line	Order Voucher not found	Order Voucher not found
31	MSG31	In line	Category not found	Category not found
32	MSG32	In line	Product not found	Product not found
33	MSG33	In line	Product Size not found	Product Size not found
34	MSG34	In line	Recipe not found	Recipe not found
35	MSG35	In line	Invalid Recipe Unit	Invalid Unit of Measurement
36	MSG36	In line	Invalid Order Item status	Invalid Order Item Status
37	MSG37	In line	Order Item not found	Order Item not found
38	MSG38	In line	Option Item not found	Option Item not found
39	MSG39	In line	Payment not found	Payment not found
40	MSG40	In line	Customer not found	Customer not found
41	MSG41	In line	Feedback not found	Feedback not found
42	MSG42	In line	Order not found	Order not found
43	MSG43	In line	Order already invoiced or paid	Order has already been invoiced or paid
44	MSG44	In line	Order has not been invoiced or paid yet	Order has not been invoiced or paid yet
45	MSG45	In line	Invalid staff status	Invalid Staff Status
46	MSG46	In line	Invalid staff type	Invalid Staff Type
47	MSG47	In line	Option not found	Option not found
48	MSG48	In line	Staff Zone not found	Staff Zone not found
49	MSG49	In line	Table not found	Table not found
50	MSG50	In line	Invalid table status	Invalid Table status
51	MSG51	In line	Table order not found	Table has no order
52	MSG52	In line	Table order missing order items	Order contains no items
53	MSG53	In line	Table order missing total price	Order Total has not been calculated

54	MSG54	In line	Invalid table capacity	Table Capacity must be greater than 0
55	MSG55	In line	Voucher not found	Voucher not found
56	MSG56	In line	Voucher Type not found	Voucher Type not found
57	MSG57	In line	Invalid total quantity for voucher type	Total Quantity must be greater than 0
58	MSG58	In line	Zone not found	Zone not found

Table 70- List of messages

IV. Software Design Document

1. System Design

1.1 System Architecture

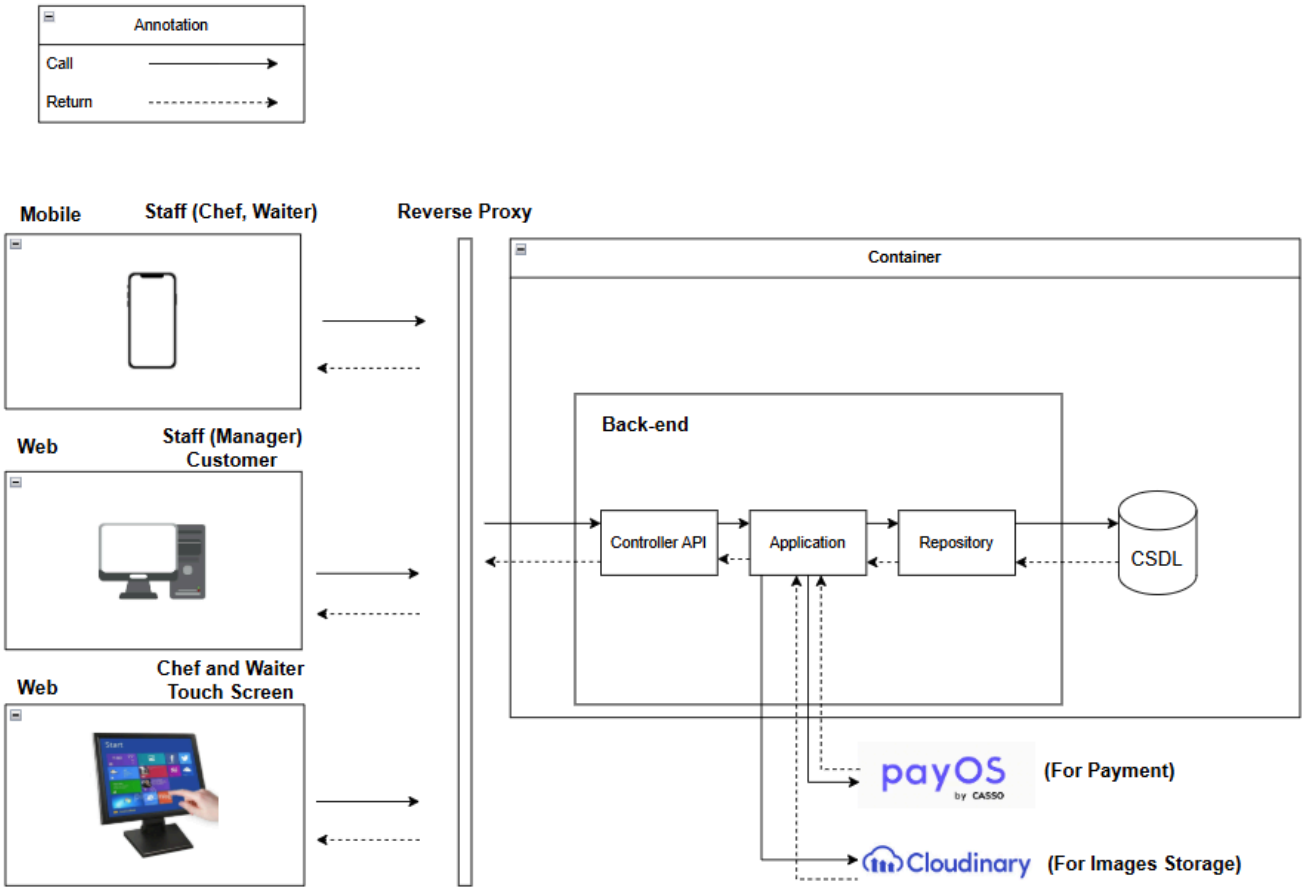


Figure 71 - System overview diagram

1.2 Package Diagram

1.2.1 Backend Package Diagram

1.2.1.1 Diagram

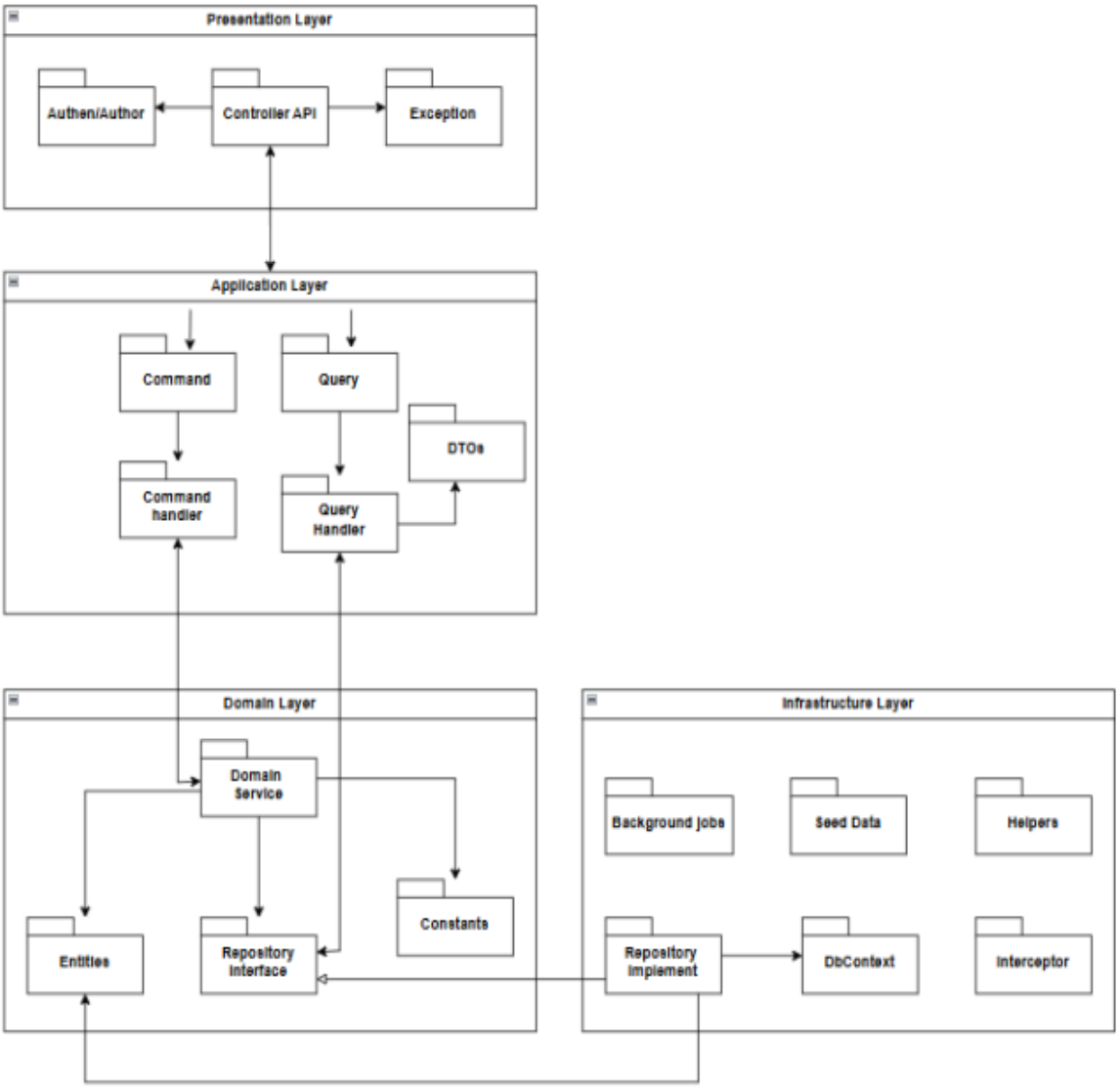


Figure 72- Backend package diagram

1.2.1.2 Descriptions

No	Package	Description
1	Controller API	Contains the API controllers responsible for handling incoming HTTP requests, delegates to handlers in the Application Layer, and returning responses to the client.

2	Authen/Author	Handles user authentication (JWT token validation) and authorization
3	Exception	Manages global error handling
4	Command / Query	Command: Represents actions that modify state (e.g., Create, Update, Delete). Query: Represents actions that retrieve data. Follows the CQRS pattern (Command Query Responsibility Segregation) to separate read and write logic.
5	Command Handler / Query Handler	Contains the business logic to process commands and queries. Helps maintain clean separation and testability.
6	DTOs (Data Transfer Objects)	Define the shape of data for input/output between the API and other layers. Help decouple the domain model from external interfaces.
7	Entities	Represents the domain objects in the system, directly mapped to tables in the database. Encapsulate business logic and rules.
8	Domain Service	Contains business logic that doesn't naturally belong to a single entity.
9	Repository Interfaces	Define contracts for data access operations Implemented in the Infrastructure Layer to maintain separation of concerns.
10	Constants	Common constant values used across the domain, like error response
11	Repository Implement	Concrete implementation of repository interfaces, typically using EF Core
12	DbContext	Manages database access and tracks changes using Entity Framework Core.
13	Interceptor	Middleware components for soft delete, auditing, etc., during DB operations.
14	Background Jobs	Handles background processing tasks like open/close register workshop, notify assign table reservation, auto assign staff zone schedule

15	Seed Data	Provides initial/default data for the system
16	Helpers	Utility classes for formatting, converting, or general-purpose operations.

Table 73- Package Back-end Descriptions

1.2.2 Website Package Diagram

1.2.2.1 Diagram

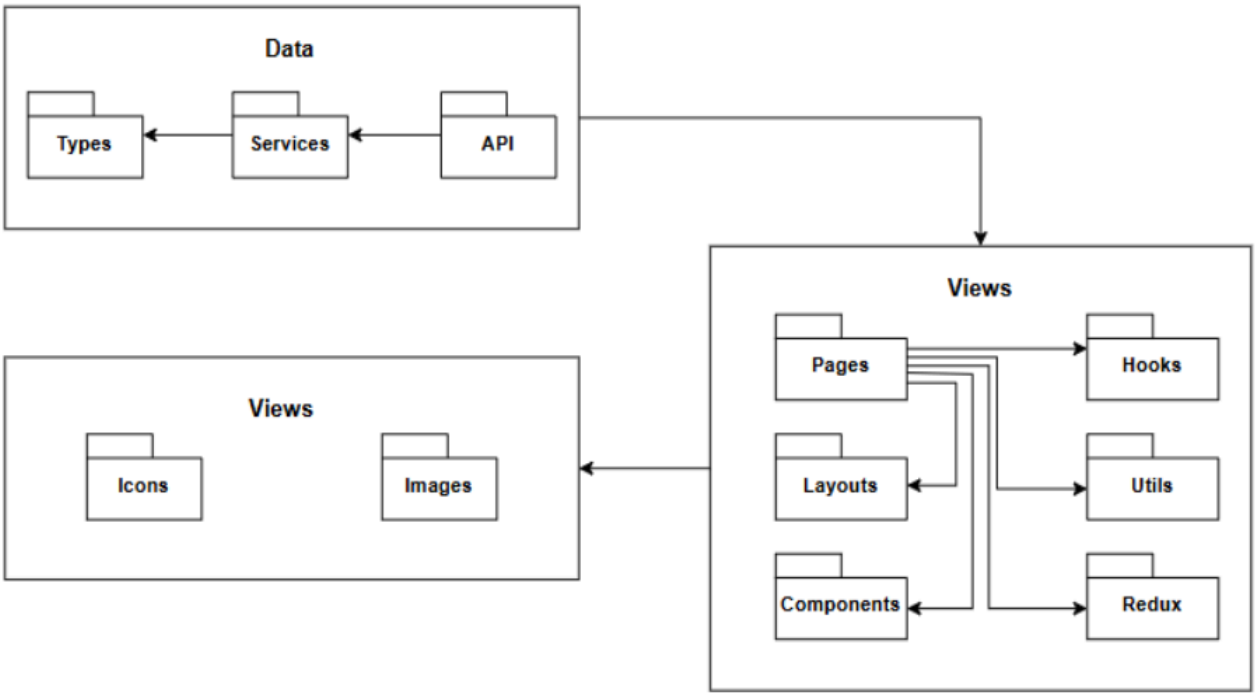


Figure 74 - Website package diagram

1.2.2.2 Descriptions

No	Package	Description
01	Types	Contains type definitions used throughout the application to ensure consistent data structure and typing.
02	Icon	Contains icon assets used across the application.
03	Images	Contains image assets used across the application.
04	Pages	Contains the different pages of the application, typically representing different routes or views in the app.

05	Hooks	Contains custom hooks to manage component logic and state management in a reusable way.
06	Layouts	Contains layout components that structure and organize the design of pages.
07	Utils	Contains utility functions and helpers that are used across the application.
08	Redux	Manages the global state of the application and provides a predictable way to handle state changes.
09	Components	Contains reusable components that can be used across various pages.
10	Api	Contains methods and logic for interacting with backend APIs and services to fetch or send data.
11	Services	Responsible for handling data fetching, communication with APIs, and managing data logic.

Table 75 - Package Front-end Descriptions

2. Database Design

2.1 Database

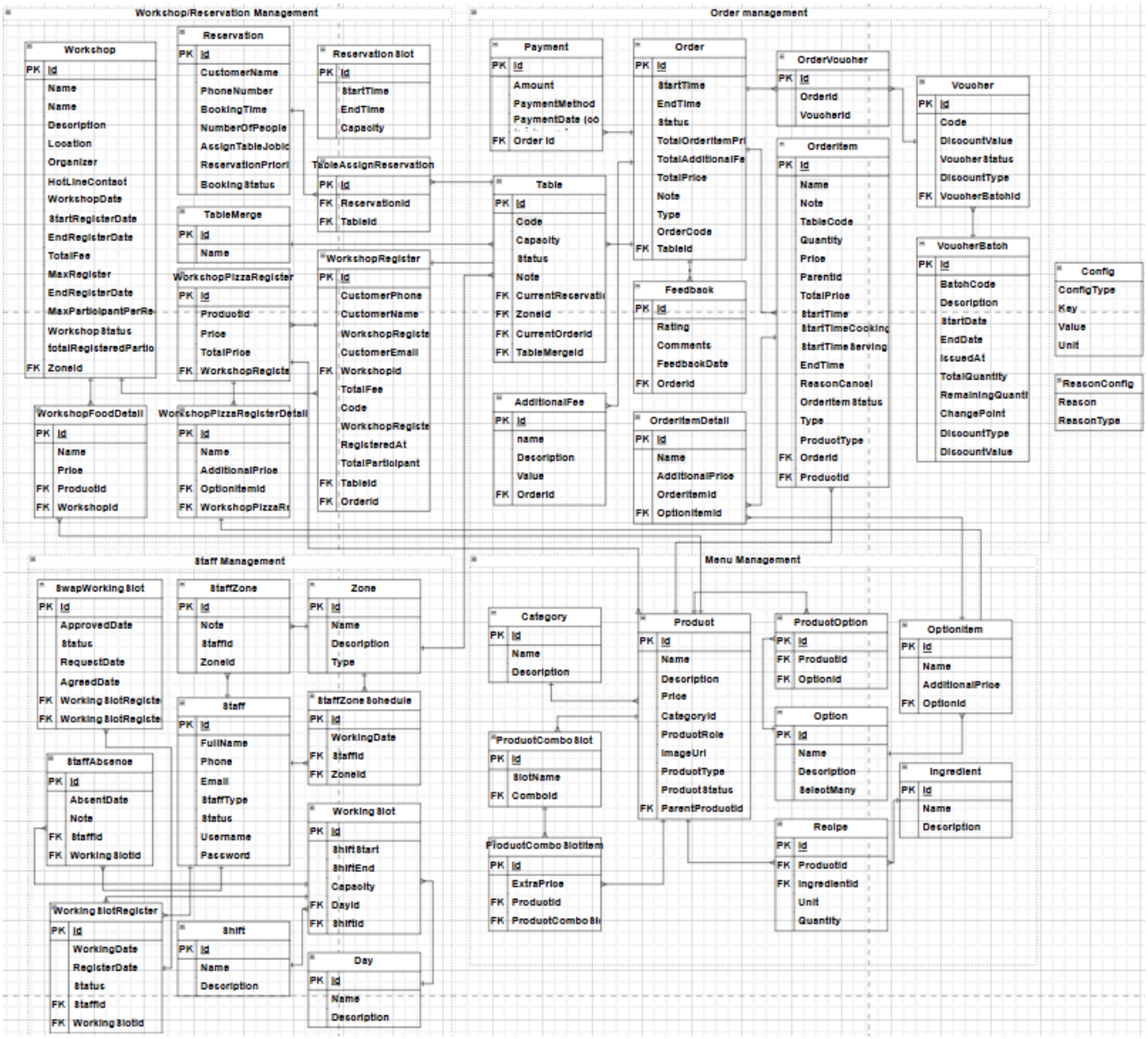


Figure 76 - Database

2.2 Data Dictionary

2.2.1 Table Workshop

Attributes	Description	Type	Required
Id	Workshop ID	UUID	Yes
Name	Workshop name	String	Yes

Description	Description	Text	No
Location	Location of workshop	String	Yes
Organizer	Organizer name	String	Yes
HotlineContact	Hotline contact	String	No
WorkshopDate	Workshop date	Date	Yes
StartRegisterDate	Registration start date	Date	Yes
EndRegisterDate	Registration end date	Date	Yes
TotalFee	Total fee	Decimal	Yes
MaxRegister	Max registrations	Integer	Yes
MaxParticipantPerRole	Max per role	Integer	No
WorkshopStatus	Status	Enum	Yes
TotalRegisteredPartic	Total registered	Integer	No
ZoneId	Reference to zone	UUID	Yes

2.2.2 Table Reservation

Attributes	Description	Type	Required
Id	Reservation ID	UUID	Yes
CustomerName	Name of the customer	String	Yes
PhoneNumber	Contact number	String	Yes
BookingTime	Time booked	Datetime	Yes
NumberOfPeople	Number of people	Integer	Yes
AssignTableJobId	Reference to job assigned	UUID	No
ReservationPriority	Priority	Integer	No
BookingStatus	Reservation status	Enum	Yes

2.2.3 Table ReservationSlot

Attributes	Description	Type	Required
Id	Slot ID	UUID	Yes

StartTime	Start time	Time	Yes
EndTime	End time	Time	Yes
Capacity	Slot capacity	Integer	Yes

2.2.4 Table TableAssignReservation

Attributes	Description	Type	Required
Id	Assign ID	UUID	Yes
ReservationId	Reference to reservation	UUID	Yes
TableId	Reference to table	UUID	Yes

2.2.5 Table TableMerge

Attributes	Description	Type	Required
Id	Table merge ID	UUID	Yes
Name	Merge group name	String	Yes

2.2.6 Table WorkshopPizzaRegister

Attributes	Description	Type	Required
Id	Register ID	UUID	Yes
ProductId	Product registered	UUID	Yes
Price	Price	Decimal	Yes
TotalPrice	Total price	Decimal	Yes
WorkshopRegisterId	Linked workshop reg.	UUID	Yes

2.2.7 Table WorkshopRegister

Attributes	Description	Type	Required
Id	Register ID	UUID	Yes
CustomerPhone	Phone number	String	Yes
CustomerName	Customer name	String	Yes
WorkshopRegisterDate	Register date	DateTime	Yes

CustomerEmail	Email address	String	No
TotalFee	Total fee	Decimal	Yes
TableId	Table reference	UUID	Yes
OrderId	Order reference	UUID	No
Code	Registration code	String	Yes
RegisteredAt	Timestamp	Datetime	Yes
TotalParticipant	Number of participants	Integer	Yes

2.2.8 Table WorkshopPizzaRegisterDetail

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
AdditionalPrice	Additional price for pizza option items at the register time	decimal	Yes
OptionItemId	Reference to selected option item	UUID	Yes
WorkshopPizzaRegisterId	Reference to pizza registration	UUID	Yes

2.2.9 Table WorkshopFoodDetail

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Name	Name of the food item	varchar	Yes
Price	Price of the item	decimal	Yes
ProductId	Linked product ID	UUID	Yes
WorkshopId	Reference to workshop	UUID	Yes

2.2.10 Table Payment

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Amount	Amount paid	decimal	Yes
PaymentMethod	Method used for payment	varchar	Yes

PaymentDate	Payment timestamp	datetime	Yes
OrderId	Linked order ID	UUID	Yes

2.2.11 Table Order

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
StartTime	Time when order started	datetime	Yes
EndTime	Time when order ended	datetime	No
Status	Order status	varchar	Yes
TotalOrderItemPrice	Total price of items	decimal	Yes
TotalAdditionalFee	Sum of additional fees	decimal	No
TotalPrice	Final total price	decimal	Yes
Note	Any notes on the order	varchar	No
Type	Type of order	varchar	Yes
OrderCode	Unique code for order	varchar	Yes
TableId	Reference to table used	UUID	No

2.2.12 Table Feedback

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Rating	Rating given	int	Yes
Comments	Additional comments	text	No
FeedbackDate	Date of feedback submitted	datetime	Yes
OrderId	Linked order ID	UUID	Yes

2.2.13 Table OrderVoucher

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes

OrderId	Reference to order	UUID	Yes
VoucherId	Reference to voucher	UUID	Yes

2.2.14 Table Voucher

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Code	Unique voucher code	varchar	Yes
DiscountValue	Discount value (percentage/amount)	decimal	Yes
VoucherStatus	Current status of the voucher	varchar	Yes
DiscountType	Type of discount	varchar	Yes
VoucherBatchId	Reference to voucher batch	UUID	Yes

2.2.15 Table OrderItem

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Name	Name of the item	varchar	Yes
Note	Additional notes	varchar	No
TableCode	Code of table served	varchar	No
Quantity	Quantity ordered	int	Yes
Price	Price per item	decimal	Yes
ParentId	Parent order item (combo etc.)	UUID	No
TotalPrice	Total item price	decimal	Yes
StartTimeCooking	Start cooking time	datetime	No
StartTimeServing	Start serving time	datetime	No
EndTime	End time of service	datetime	No
ReasonCancel	Reason for cancellation	varchar	No
OrderItemStatus	Status of item	varchar	Yes
Type	Type/category of item	varchar	Yes

ProductType	Type of product	varchar	Yes
OrderId	Linked order ID	UUID	Yes
ProductId	Linked product ID	UUID	No

2.2.16 Table OrderItemDetail

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Name	Detail name	varchar	Yes
AdditionalPrice	Additional price	decimal	Yes
OrderItemId	Reference to order item	UUID	Yes
OptionItemId	Reference to selected option	UUID	Yes

2.2.17 Table VoucherBatch

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
BatchCode	Unique batch code	varchar	Yes
Description	Description of the batch	text	No
StartDate	Start validity date	datetime	Yes
EndDate	End validity date	datetime	Yes
IssuedAt	Issuance timestamp	datetime	No
TotalQuantity	Total vouchers created	int	Yes
RemainingQuantity	Remaining vouchers available	int	Yes
DiscountType	Type of discount	varchar	Yes
DiscountValue	Discount value	decimal	Yes

2.2.18 Table Config

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes

ConfigType	Type of configuration	varchar	Yes
Key	Config key	varchar	Yes
Value	Config value	varchar	Yes
Unit	Unit for the config value	varchar	No

2.2.19 Table ReasonConfig

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Reason	Reason or message	varchar	Yes
ReasonType	Type/category of reason	varchar	Yes

2.2.20 Table SwapWorkingSlot

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
ApprovedDate	Date the swap was approved	datetime	No
Status	Current status of the swap	varchar	Yes
RequestDate	Date the request was made	datetime	Yes
AgreedDate	Date both parties agreed	datetime	No
WorkingSlotRegisterId	Reference to working slot register	int	Yes
WorkingSlotRegisterSwapId	Reference to target slot	UUID	Yes

2.2.21 Table StaffAbsence

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
AbsentDate	Date of absence	datetime	Yes
Note	Reason or note	varchar	No
StaffId	Reference to staff	UUID	Yes

WorkingSlotId	Reference to working slot	UUID	Yes

2.2.22 Table WorkingSlotRegister

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
WorkingDate	Date of working	date	Yes
RegisterDate	Date registered	datetime	Yes
Status	Registration status	varchar	Yes
StaffId	Reference to staff	UUID	Yes
WorkingSlotId	Reference to slot	UUID	Yes

2.2.23 Table StaffZone

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Note	Optional note	varchar	No
StaffId	Reference to staff	UUID	Yes
ZoneId	Reference to zone	UUID	Yes

2.2.24 Table Zone

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Name	Zone name	varchar	Yes
Description	Description of zone	varchar	No
Type	Zone type (Hot Kitchen, Cold Kitchen)	varchar	Yes

2.2.25 Table StaffZoneSchedule

Attributes	Description	Type	Required

Id	Unique identifier	UUID	Yes
WorkingDate	Date of scheduled work	date	Yes
StaffId	Reference to staff	UUID	Yes
ZoneId	Reference to zone	UUID	Yes
WorkingSlotId	Reference to working slot	UUID	Yes

2.2.26 Table Staff

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
FullName	Full name of the staff	varchar	Yes
Phone	Contact phone number	varchar	Yes
Email	Email address	varchar	Yes
StaffType	Role/type of staff	varchar	Yes
Status	Current status	varchar	Yes
Username	Login username	varchar	Yes
Password	Login password (hashed)	varchar	Yes

2.2.27 Table Shift

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Name	Shift name	varchar	Yes
Description	Shift description	varchar	No

2.2.28 Table WorkingSlot

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
ShiftStart	Start time of shift	time	Yes
ShiftEnd	End time of shift	time	Yes

Capacity	Number of slots available	int	Yes
DayId	Day of the week	UUID	Yes
ShiftId	Reference to shift	UUID	Yes

2.2.29 Table Day

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Name	Name of the day	varchar	Yes
Description	Description	varchar	No

2.2.30 Table Category

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Name	Category name	varchar	Yes
Description	Category description	varchar	No

2.2.31 Table Product

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Name	Product name	varchar	Yes
Description	Description	text	No
Price	Base price	decimal	Yes
CategoryId	Product category	UUID	Yes
ProductRole	Role of product (e.g., main)	varchar	No
ImageUrl	Image link	varchar	No
ProductType	Type of product	varchar	Yes
ProductStatus	Current status	varchar	Yes
ParentProductId	Parent product (for combo)	UUID	No

2.2.32 Table ProductOption

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
ProductId	Reference to product	UUID	Yes
OptionId	Reference to option	UUID	Yes

2.2.33 Table Option

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Name	Option name	varchar	Yes
Description	Option description	varchar	No
SelectMany	Allow multiple selection	boolean	Yes

2.2.34 Table OptionItem

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
Name	Option item name	varchar	Yes
AdditionalPrice	Extra cost if selected	decimal	Yes
OptionId	Belongs to which option	UUID	Yes

2.2.35 Table ProductComboSlot

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
SlotName	Name of slot in the combo	varchar	Yes
ComboId	Reference to product combo	UUID	Yes

2.2.36 Table ProductComboSlotItem

Attributes	Description	Type	Required

Id	Unique identifier	UUID	Yes
ExtraPrice	Extra charge for the combo item	decimal	Yes
ProductId	Linked product	UUID	Yes
ProductComboSlotId	Combo slot reference	UUID	Yes

2.2.37 Table Recipe

Attributes	Description	Type	Required
Id	Unique identifier	UUID	Yes
ProductId	Product reference	int	Yes
IngredientId	Ingredient reference	UUID	Yes
Unit	Unit of measurement	varchar	Yes
Quantity	Quantity needed	float	Yes

2.2.38 Table Table

Attributes	Description	Type	Required
Id	Unique identifier for the table	UUID	Yes
Code	Table code (e.g., T01, T02)	String	Yes
Capacity	Maximum number of people per table	Integer	Yes
Status	Current status (e.g., Available, Reserved)	String/Enum	Yes
Note	Additional information	String	No
CurrentReservationId	Currently assigned reservation	UUID	No
ZoneId	Zone this table belongs to	UUID	Yes
CurrentOrderId	Currently assigned order	UUID	No
TableMergedId	If merged, ID of the merged table group	UUID	No

2.2.39 Table Ingredient

Attributes	Description	Type	Required
Id	Unique identifier for ingredient	UUID	Yes

Name	Ingredient name	String	Yes
Description	Description or note	String	No

2.2.40 Table AdditionalFee

Attributes	Description	Type	Required
Id	Unique identifier for the fee	UUID	Yes
Name	Fee name (e.g., Service Charge)	String	Yes
Description	Description of the fee	String	No
Value	Amount to be added	Decimal	Yes
OrderId	Associated order	UUID	Yes

3. Detailed Design

3.1 <Customer> Create Reservation

3.1.1 Class Diagram

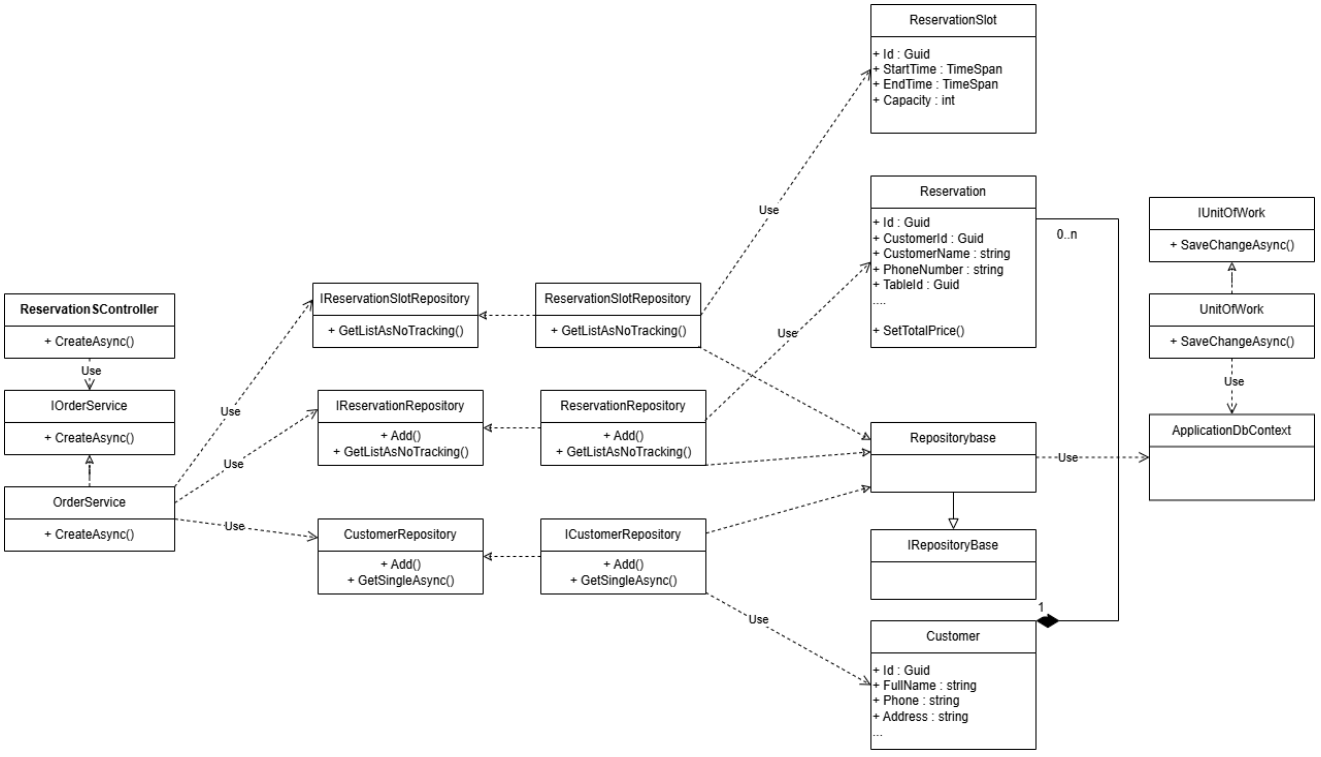


Figure 77- Create Reservation class diagram

3.1.2 Sequence diagram

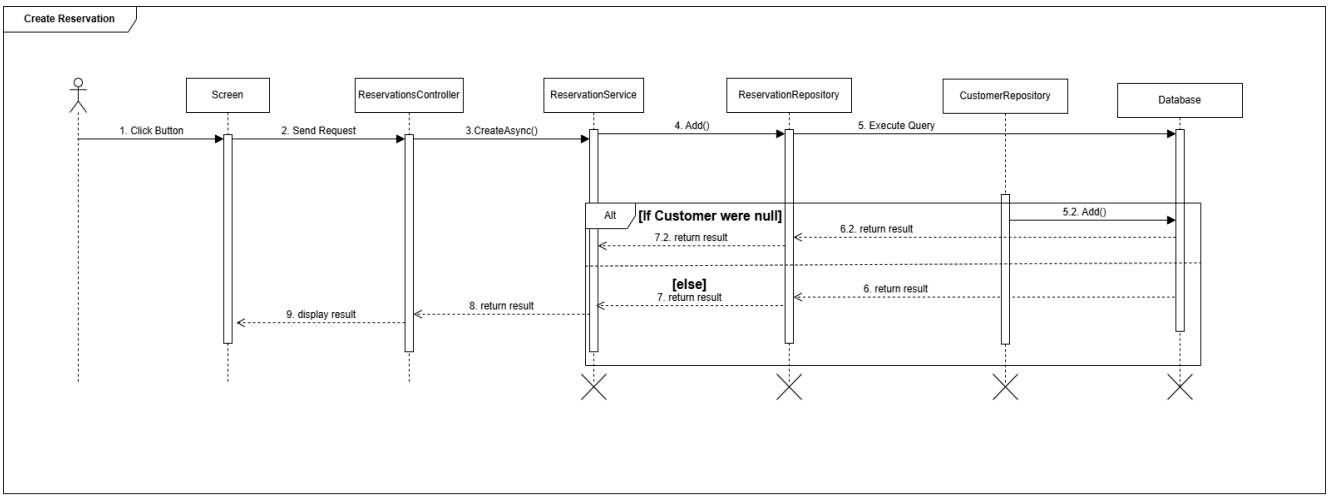


Figure 78 - Create Reservation sequence diagram

3.1.3 Class Specification

No	Class	Description
1	ReservationsController	This class handles HTTP requests for creating reservations
2	IOrderService	This is an interface defining the CreateAsync method for reservation logic
3	OrderService	This class implements business logic to create reservations and related records
4	IReservationSlotRepository	This is an interface for retrieving available reservation slots
5	ReservationSlotRepository	This class manages data access for reservation slot entities
6	IReservationRepository	This is an interface for adding and retrieving reservation records
7	ReservationRepository	This class handles data access for reservation entities
8	ICustomerRepository	This is an interface for adding and retrieving customer records
9	CustomerRepository	This class manages data access for customer entities
10	IUnitOfWork	This interface defines SaveChangeAsync for transaction management
11	UnitOfWork	This class implements the unit-of-work pattern for committing database changes
12	IRepositoryBase	This is a base interface for shared repository operations
13	RepositoryBase	This class provides common data access logic for repositories
14	ReservationSlot	This class represents a time slot available for reservations
15	Reservation	This class defines reservation details including customer, table, and pricing
16	Customer	This class represents a customer with basic contact and identity info

17	ApplicationDbContext	This class manages the EF Core database context and configuration

Figure 79 - Create Reservation Class Specification

3.2 <Staff> Assign Table to Reservation

3.2.1 Class Diagram

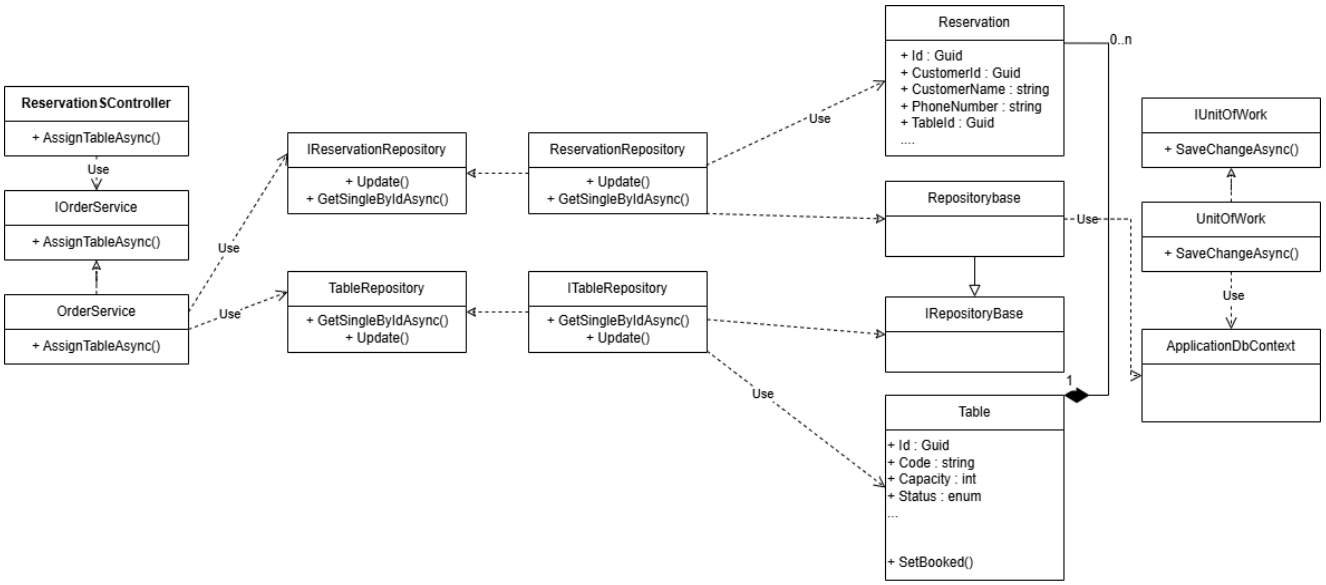


Figure 80 - Assign Table to Reservation class diagram

3.2.2 Sequence diagram

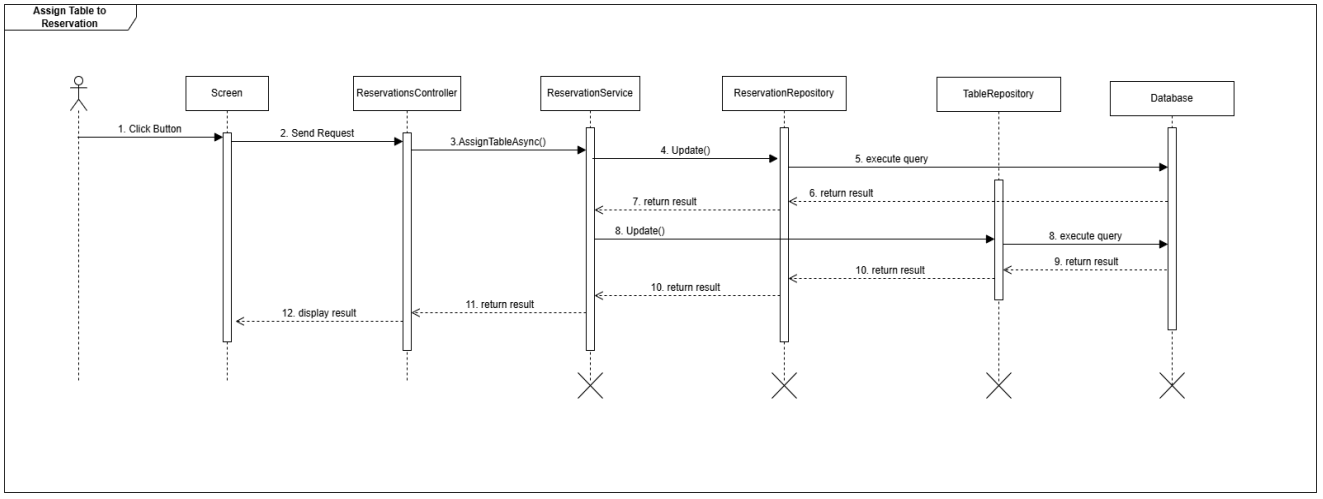


Figure 81- Assign Table to Reservation sequence diagram

3.2.3 Class Specification

No	Class	Description
1	ReservationsController	This class handles HTTP requests to assign tables to reservations
2	IOrderService	This is an interface that defines the AssignTableAsync method
3	OrderService	This class implements the logic for assigning a table to a reservation
4	IReservationRepository	This is an interface for retrieving and updating reservation data
5	ReservationRepository	This class manages data access for reservation entities
6	ITableRepository	This is an interface for retrieving and updating table records
7	TableRepository	This class handles data access for table entities
8	IUnitOfWork	This interface defines SaveChangeAsync to commit data transactions
9	UnitOfWork	This class implements transactional logic for saving data changes
10	IRepositoryBase	This is a base interface for shared update operations
11	RepositoryBase	This class provides common data access methods shared by repositories
12	Reservation	This class represents a reservation entity, including customer and table info
13	Table	This class represents a dining table and includes the SetBooked method to mark it as reserved
14	ApplicationDbContext	This class manages the Entity Framework Core database context

Figure 82- Assign Table to Reservation Class Specification

3.3 <Manager> Check in Reservation

3.3.1 Class Diagram

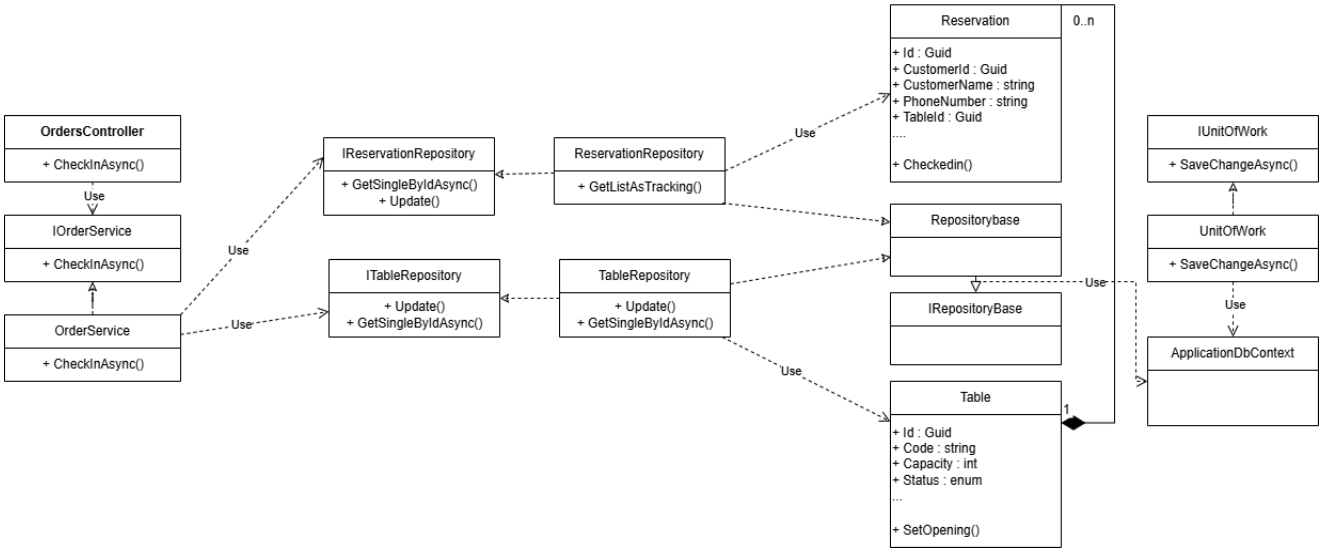


Figure 83- Check in Reservation class diagram

3.3.2 Sequence diagram

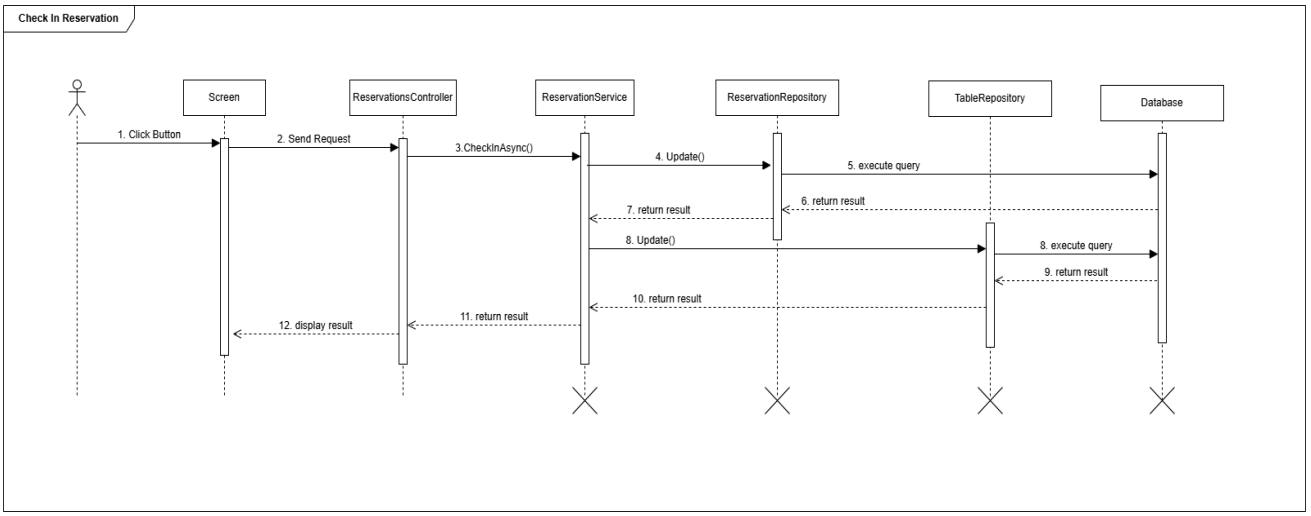


Figure 84 - Check in Reservation sequence diagram

3.3.3 Class Specification

No	Class	Description
1	OrdersController	This class handles HTTP requests to check in a customer at their reserved table
2	IOrderService	This is an interface that defines the CheckInAsync method
3	OrderService	This class implements the logic for customer check-in and table assignment
4	IReservationRepository	This is an interface for retrieving and updating reservation data
5	ReservationRepository	This class manages data access for reservation records
6	ITableRepository	This is an interface for retrieving and updating table data
7	TableRepository	This class handles data access for table entities
8	IUnitOfWork	This interface defines SaveChangeAsync for transactional data operations
9	UnitOfWork	This class commits transactional changes to the database
10	IRepositoryBase	This is a generic repository interface with common update logic
11	RepositoryBase	This class provides reusable base logic for all repository implementations
12	Reservation	This class represents a reservation entity with customer and table info, and a CheckedIn method
13	Table	This class defines a table entity with properties like code, capacity, status, and SetOpening method
14	ApplicationDbContext	This class manages the database context using EF Core

Figure 85 - Check in Reservation Class Specification

3.4 <Customer> Create Order

3.3.1 Class Diagram

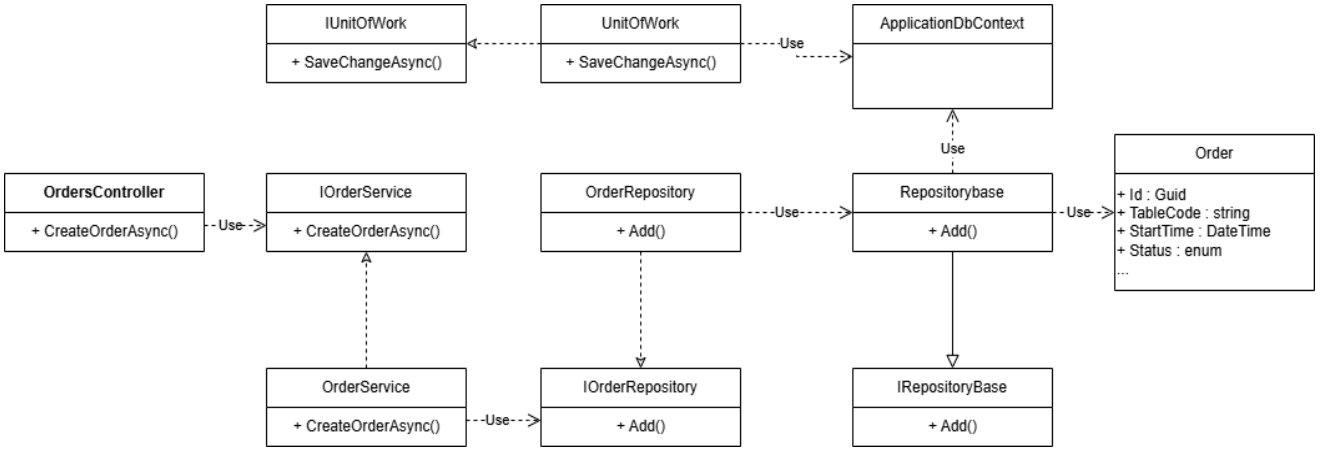


Figure 86- Create Order class diagram

3.3.2 Sequence diagram

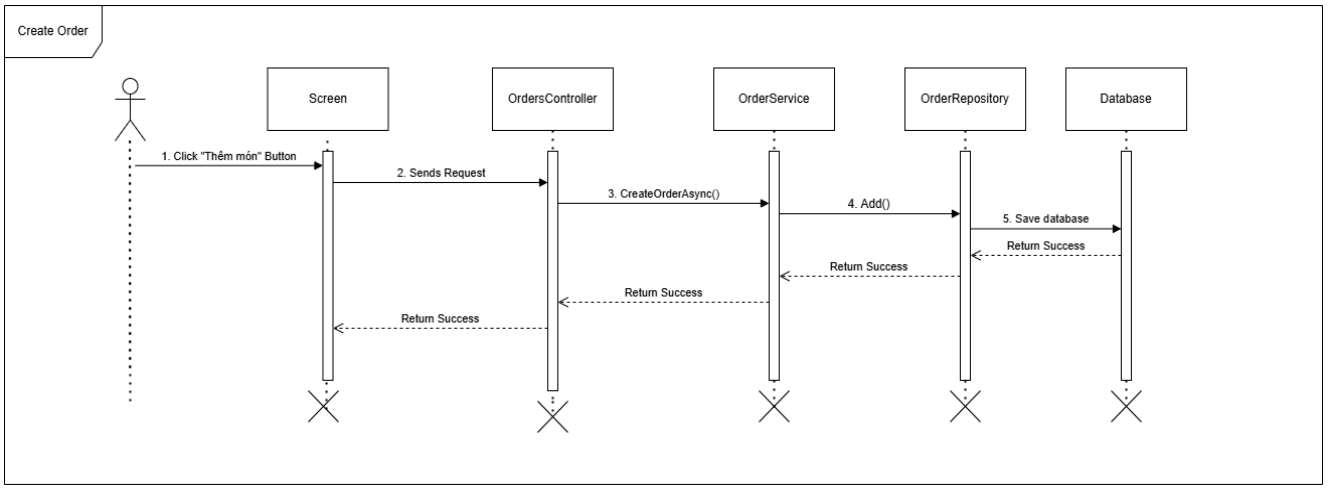


Figure 87- Create Order sequence diagram

3.3.3 Class Specification

No	Class	Description
1	OrdersController	This class handles HTTP requests to create new orders
2	IOrderService	This is an interface defining the CreateOrderAsync method
3	OrderService	This class implements the logic for creating new orders
4	IOrderRepository	This is an interface for adding new order records
5	OrderRepository	This class handles data access for persisting order entities
6	IUnitOfWork	This interface provides SaveChangeAsync for transactional

		operations
7	UnitOfWork	This class implements SaveChangeAsync to commit data changes atomically
8	IRepositoryBase	This is a shared interface for common repository operations
9	RepositoryBase	This class implements base data operations including add
10	Order	This class represents a customer order with attributes like TableCode, StartTime, and Status
11	ApplicationDbContext	This class manages the Entity Framework Core context for database interaction

Figure 88- Create Order Class Specification

3.5 <Manager> Add Food to Order

3.5.1 Class Diagram

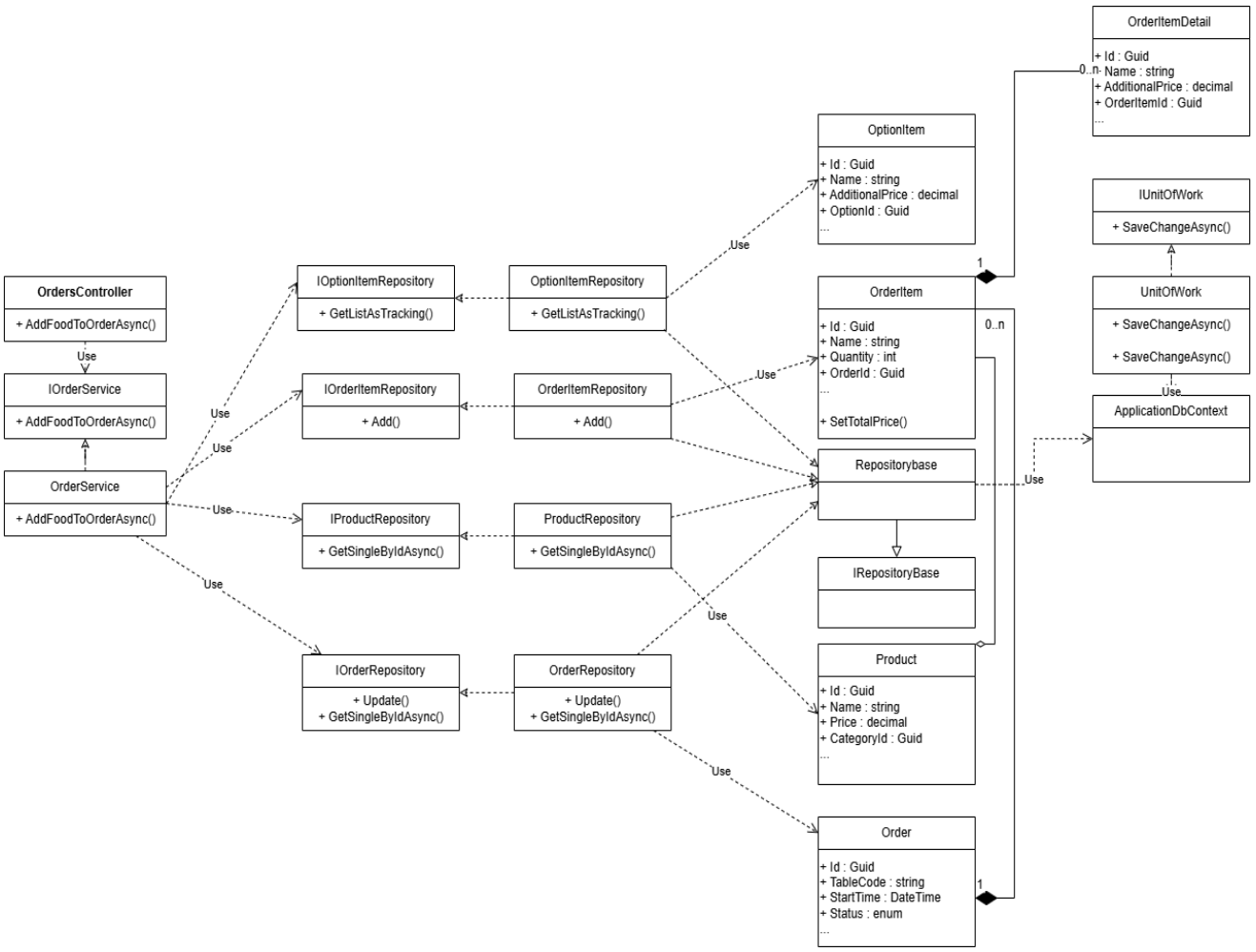


Figure 89- Add Food to Order class diagram

3.5.2 Sequence Diagram

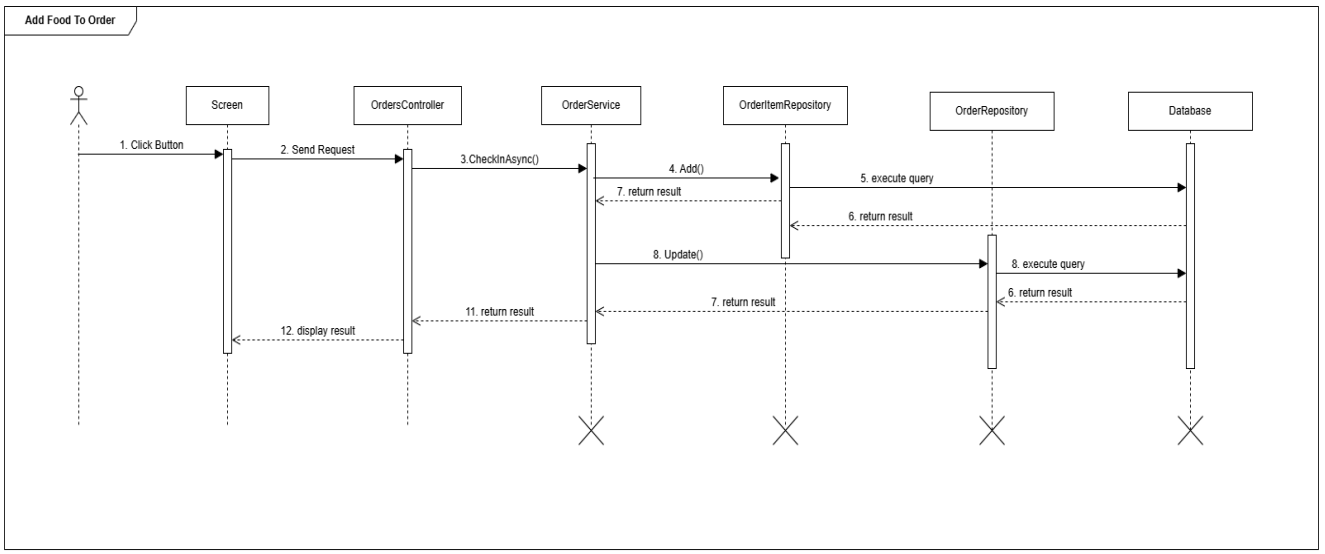


Figure 90- Add Food to Order sequence diagram

3.5.3 Class Specification

No	Class	Description
1	OrdersController	This class handles HTTP requests to add food items to an order
2	IOrderService	This is an interface defining the AddFoodToOrderAsync method
3	OrderService	This class implements the business logic for adding food to an order
4	IOptionItemRepository	This is an interface for retrieving option items associated with a food product
5	OptionItemRepository	This class handles data access for option item records
6	IOrderItemRepository	This is an interface for adding new order items
7	OrderItemRepository	This class manages data access for order item entities
8	IProductRepository	This is an interface for retrieving product data
9	ProductRepository	This class handles data access for product records
10	IOrderRepository	This is an interface for retrieving and updating order data
11	OrderRepository	This class manages data access for order records
12	IUnitOfWork	This interface defines SaveChangeAsync to manage transaction commits
13	UnitOfWork	This class implements SaveChangeAsync to persist changes
14	IRepositoryBase	This is a base interface for update operations
15	RepositoryBase	This class provides common repository logic shared across implementations

16	OptionItem	This class represents a selectable option with an additional price tied to a product
17	OrderItem	This class represents an item in an order and includes pricing logic
18	OrderItemDetail	This class represents detailed customizations for an order item
19	Product	This class defines a food product with name, price, and category ID
20	Order	This class represents an order placed by a customer, including status and time
21	ApplicationDbContext	This class manages the EF Core database context and model persistence

Figure 91- Add Food to Order Class Specification

3.6 <Customer> Check Out Order

3.6.1 Class Diagram

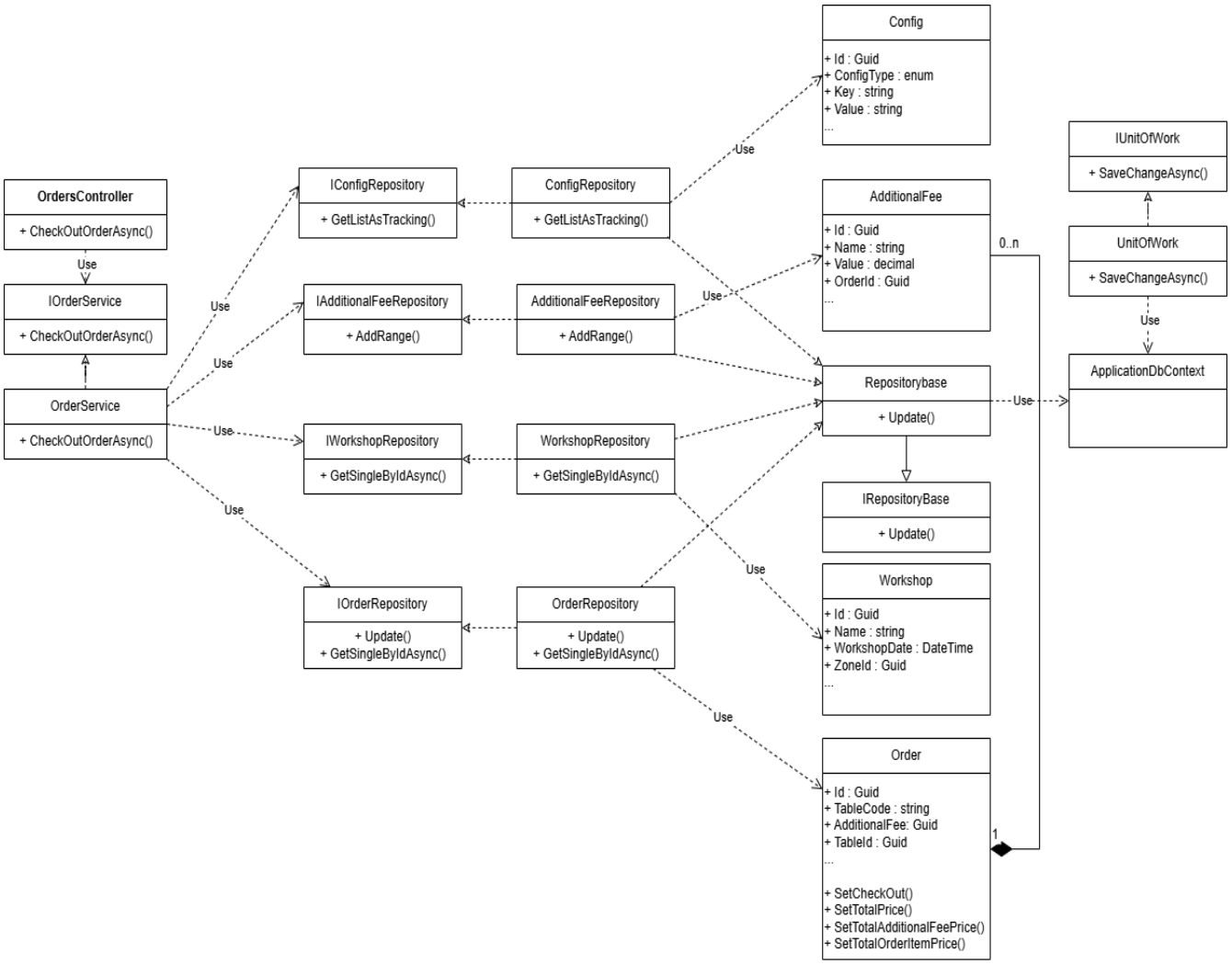


Figure 92- Check Out Order class diagram

3.6.2 Sequence Diagram

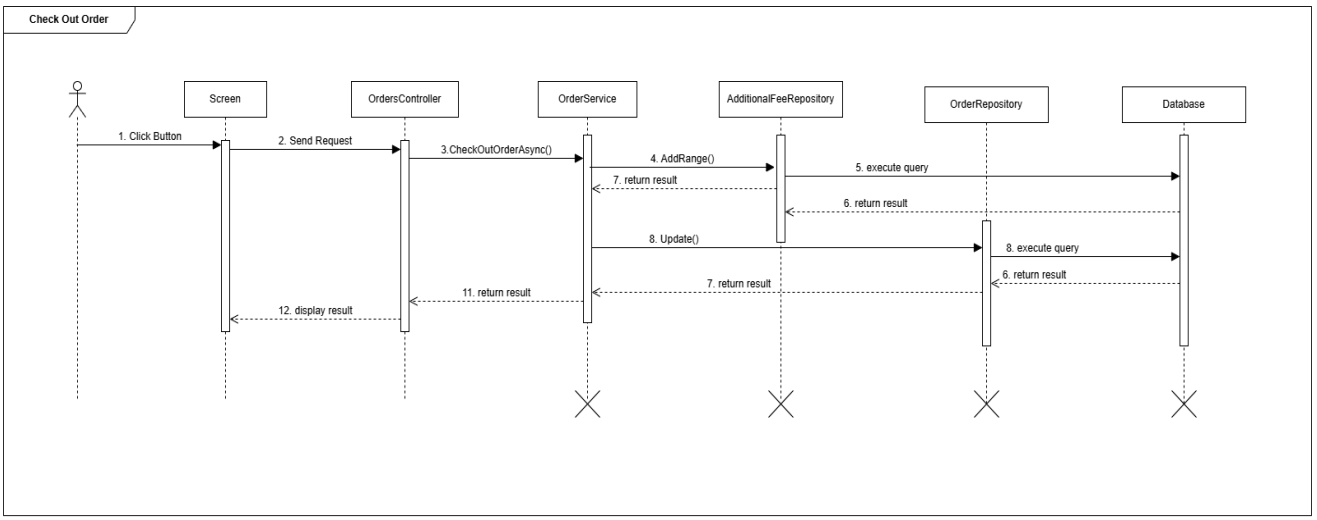


Figure 93 - Check Out Order sequence diagram

3.6.3 Class Specification

No	Class	Description
1	OrdersController	This class handles HTTP requests to check out an order
2	IOrderService	This is an interface that defines the CheckOutOrderAsync method
3	OrderService	This class contains the business logic to handle order checkout processes
4	IConfigRepository	This is an interface for retrieving configuration data
5	ConfigRepository	This class handles data access for configuration entities
6	IAdditionalFeeRepository	This is an interface for adding multiple additional fee records
7	AdditionalFeeRepository	This class manages data access for additional fee entries
8	IWorkshopRepository	This is an interface for retrieving workshop data
9	WorkshopRepository	This class handles data access for workshop entities
10	IOrderRepository	This is an interface for retrieving and updating order data
11	OrderRepository	This class manages data access for order entities
12	IUnitOfWork	This interface defines SaveChangeAsync for committing transactions
13	UnitOfWork	This class implements transaction saving logic for entity changes
14	IRepositoryBase	This is a shared interface for basic repository operations
15	RepositoryBase	This class provides reusable update functionality for repositories
16	Config	This class represents configuration settings with key-value pairs

17	AdditionalFee	This class defines an additional charge associated with a specific order
18	Workshop	This class represents a workshop session, including name, date, and zone
19	Order	This class represents a customer order and includes checkout-related methods
20	ApplicationDbContext	This class manages the EF Core database context and connection logic

Figure 94- Check Out Order Class Specification

3.7 <Customer> Cooking Order Item

3.7.1 Class Diagram

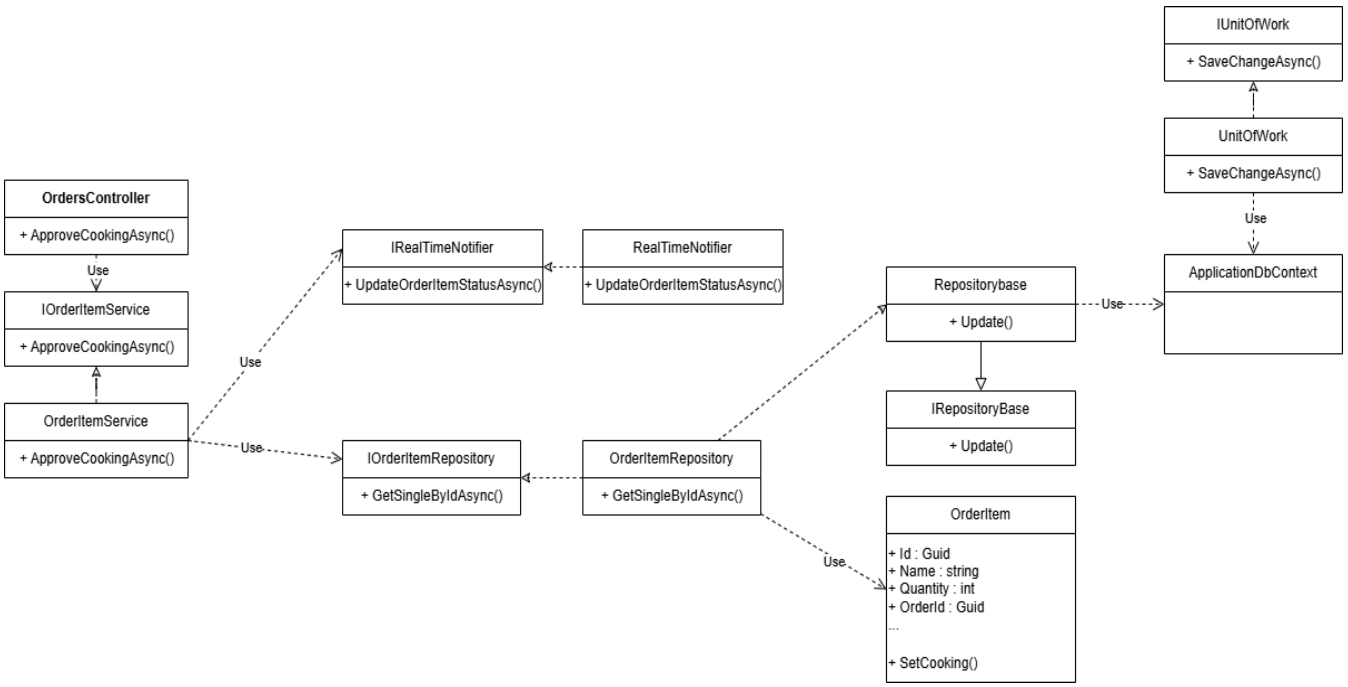


Figure 95-Cooking Order Item class diagram

3.7.2 Sequence Diagram

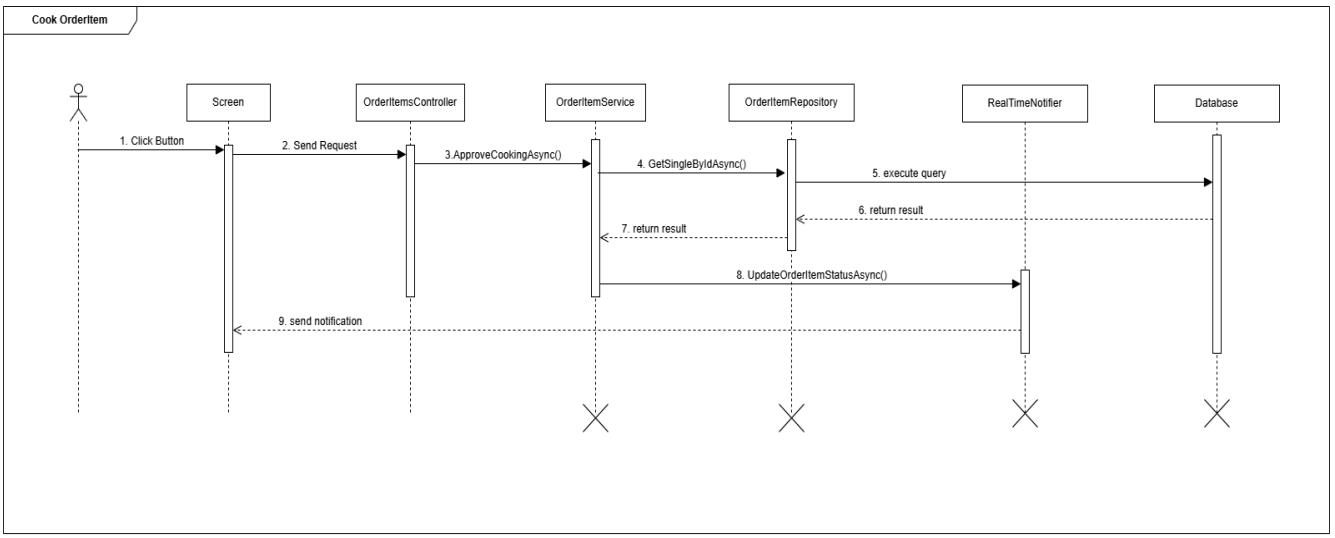


Figure 96 - Cooking Order Item sequence diagram

3.7.3 Class Specification

No	Class	Description
1	OrdersController	This class handles HTTP requests to mark an order item as done cooking
2	IOrderItemService	This is an interface defining the method DoneCookingAsync
3	OrderItemService	This class implements business logic to mark order items as cooked
4	IOrderItemRepository	This is an interface for retrieving specific order item data
5	OrderItemRepository	This class handles data access for order item entities
6	IRealTimeNotifier	This is an interface for updating the order item status in real-time
7	RealTimeNotifier	This class implements real-time status update notifications for order items
8	IUnitOfWork	This interface defines SaveChangeAsync for transaction handling
9	UnitOfWork	This class implements transaction commits for entity changes
10	IRepositoryBase	This is a generic base interface for common repository operations
11	RepositoryBase	This class provides shared update functionality for repository classes
12	OrderItem	This class represents an order item with properties and the SetServing method
13	ApplicationDbContext	This class manages the EF Core database context and persistence layer

Figure 97- Cooking Order Item Class Specification

3.8 <Chef> Done Cooking OrderItem

3.8.1 Class Diagram

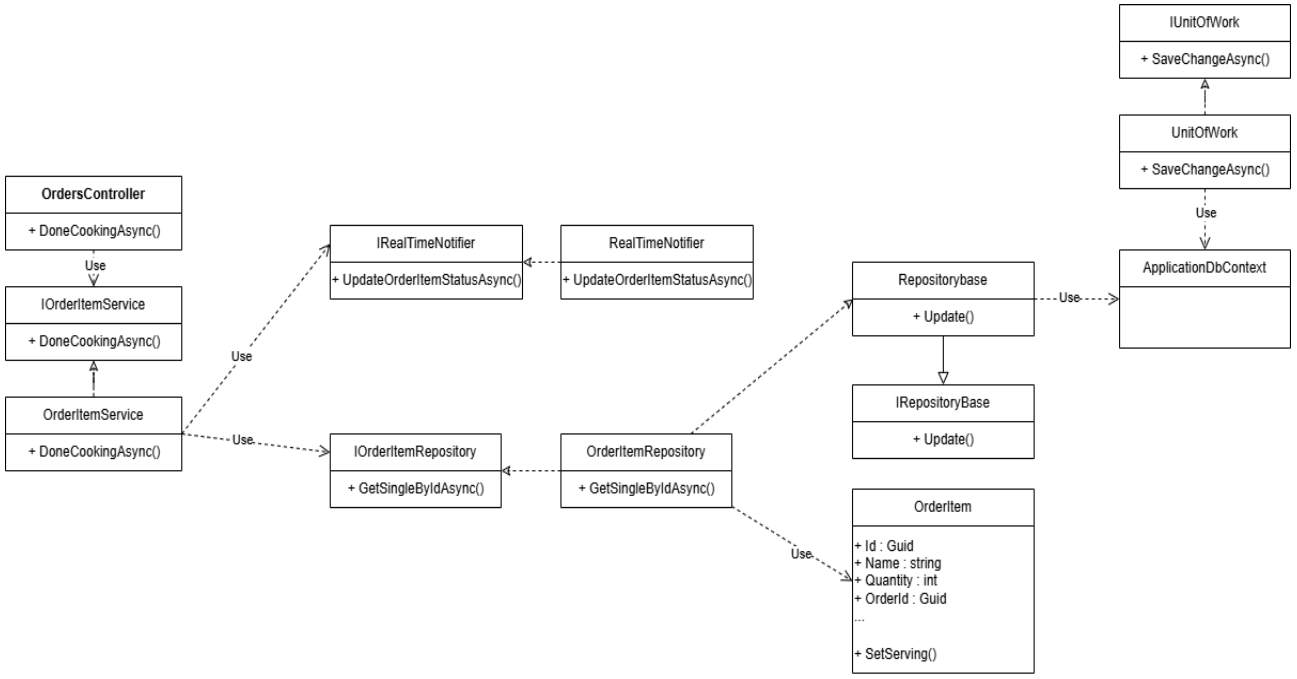


Figure 98 - Done Cooking OrderItem activity diagram

3.8.2 Sequence Diagram

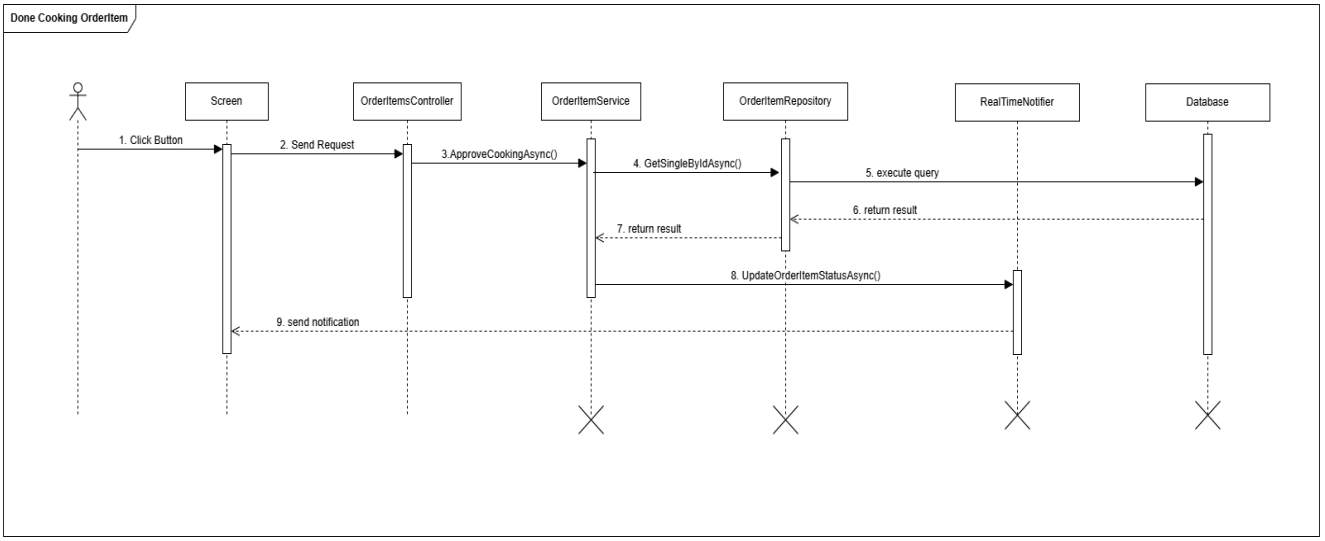


Figure 99 - Done Cooking OrderItem sequence diagram

3.8.3 Class Specification

No	Class	Description
1	OrdersController	This class handles HTTP requests to mark an order item as done cooking

2	IOrderItemService	This is an interface defining the method DoneCookingAsync
3	OrderItemService	This class implements business logic to mark order items as cooked
4	IOrderItemRepository	This is an interface for retrieving specific order item data
5	OrderItemRepository	This class handles data access for order item entities
6	IRealTimeNotifier	This is an interface for updating the order item status in real-time
7	RealTimeNotifier	This class implements real-time status update notifications for order items
8	IUnitOfWork	This interface defines SaveChangeAsync for transaction handling
9	UnitOfWork	This class implements transaction commits for entity changes
10	IRepositoryBase	This is a generic base interface for common repository operations
11	RepositoryBase	This class provides shared update functionality for repository classes
12	OrderItem	This class represents an order item with properties and the SetServing method
13	ApplicationDbContext	This class manages the EF Core database context and persistence layer

Figure 100 - Done Cooking OrderItem Class Specification

3.9 <Manager> Done Serving OrderItem

3.9.1 Class Diagram

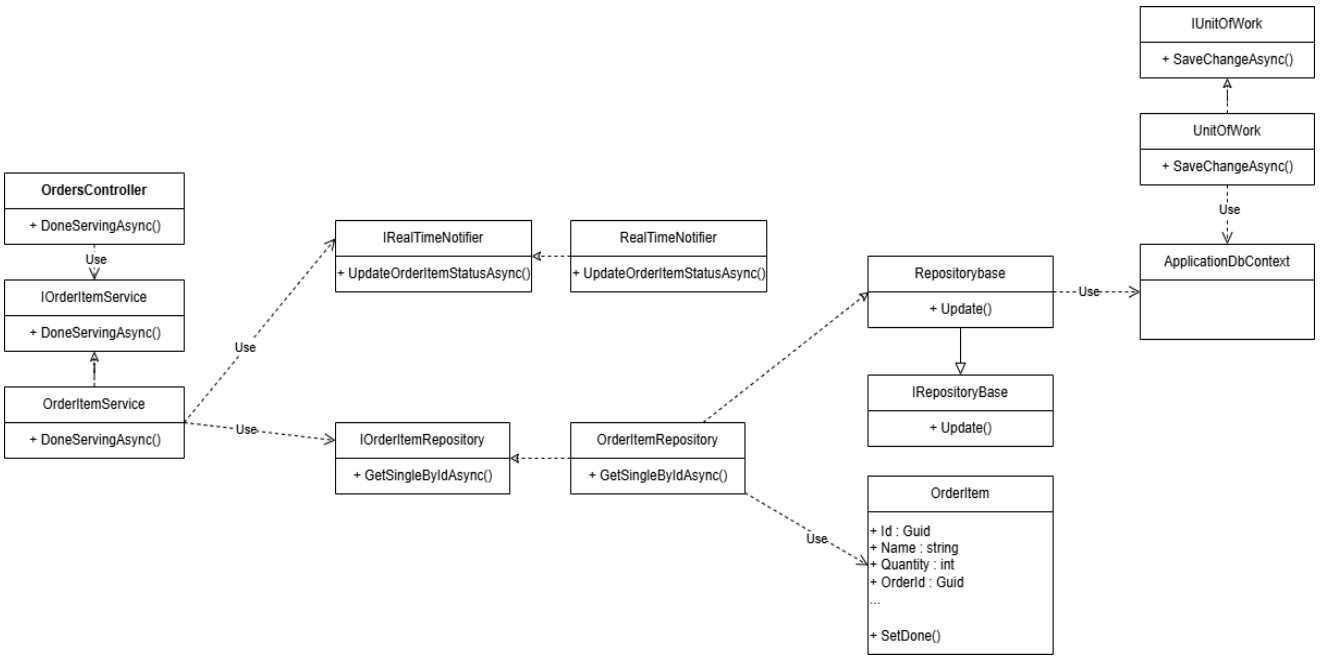


Figure 101 - Done Serving OrderItem class diagram

3.9.2 Sequence Diagram

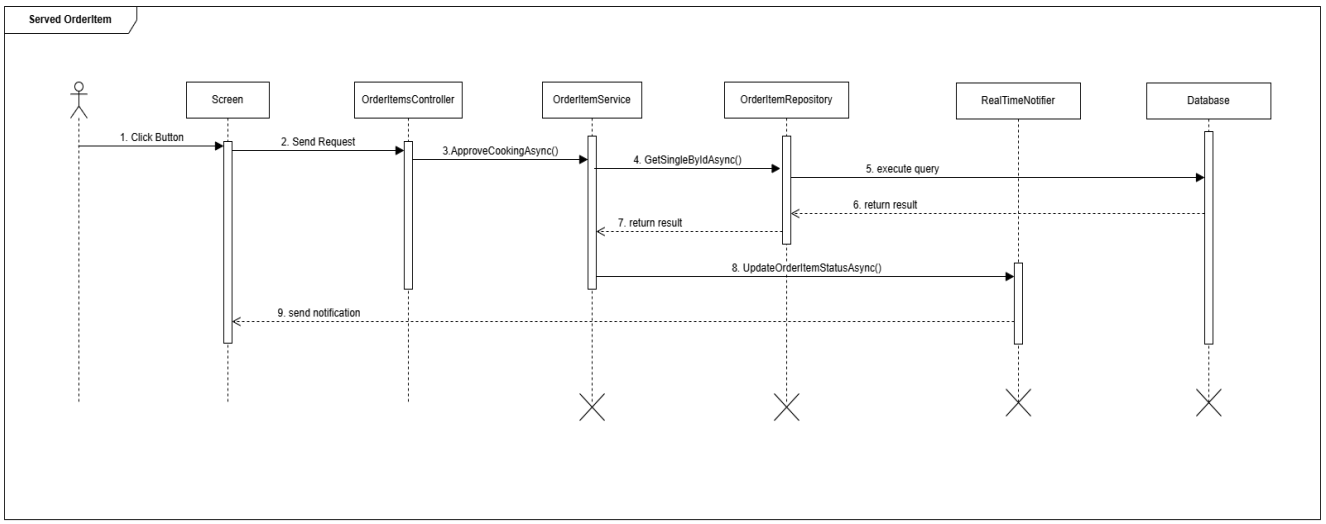


Figure 102 - Done Serving OrderItem sequence diagram

3.9.3 Class Specification

No	Class	Description
1	OrdersController	This class handles HTTP requests to mark an order item as done serving
2	IOrderItemService	This is an interface defining the method DoneServingAsync
3	OrderItemService	This class implements business logic to handle completion of serving an order item
4	IOrderItemRepository	This is an interface for retrieving specific order item data
5	OrderItemRepository	This class handles data access for order item entities
6	IRealTimeNotifier	This is an interface for updating the order item status in real-time
7	RealTimeNotifier	This class implements real-time updates to notify status changes for order items
8	IUnitOfWork	This interface defines SaveChangeAsync for transaction control
9	UnitOfWork	This class commits transactional changes to the database
10	IRepositoryBase	This is a generic interface for shared update operations
11	RepositoryBase	This class provides base repository logic reused by others
12	OrderItem	This class represents an individual item within an order and includes the SetDone method
13	ApplicationDbContext	This class manages EF Core database context and entity configurations

Figure 103 - Done Serving OrderItem Class Specification

3.10 <Staff> Create Payment QR

3.10.1 Class Diagram

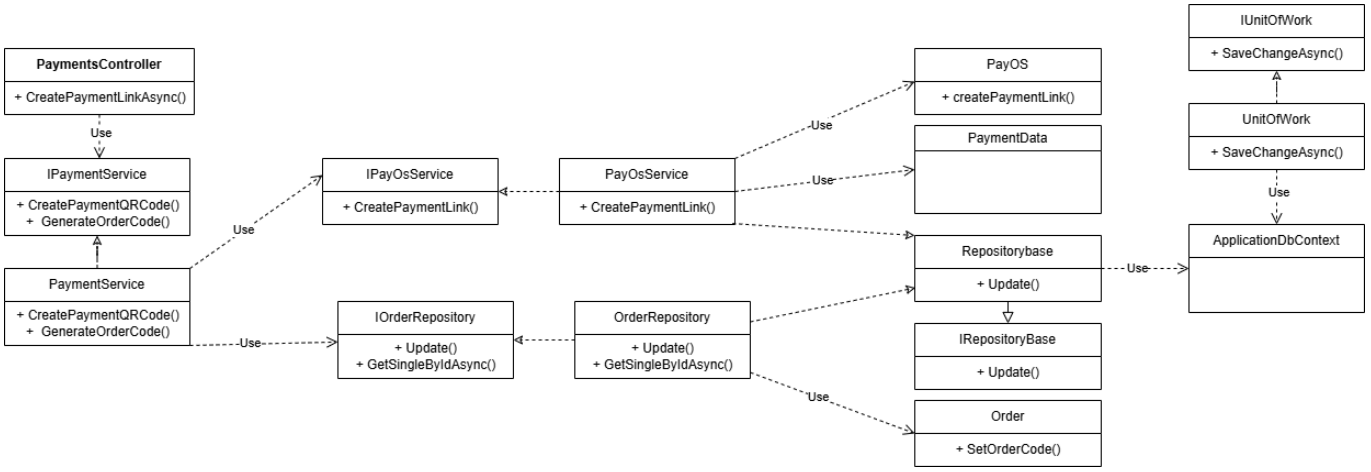


Figure 104 - Create Payment QR class diagram

3.10.2 Sequence Diagram

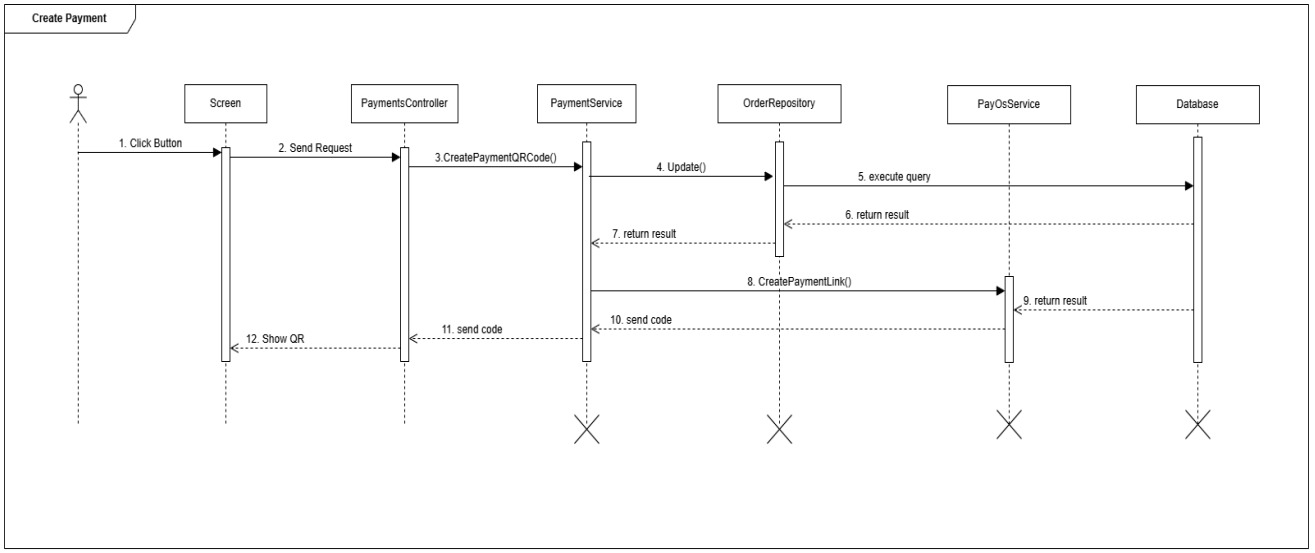


Figure 105 - Create Payment QR sequence diagram

3.10.3 Class Specification

No	Class	Description
1	PaymentsController	This class handles HTTP requests for generating QR code payment links
2	IPaymentService	This is an interface defining methods for generating payment QR codes and order codes
3	PaymentService	This class implements business logic for QR code generation and order code creation

4	IPayOsService	This is an interface for interacting with the PayOS payment gateway
5	PayOsService	This class integrates with PayOS and implements the payment link creation logic
6	PayOS	This class provides the payment link creation function from the PayOS provider
7	PaymentData	This class likely holds temporary or structured data passed between payment services
8	IOrderRepository	This is an interface for retrieving and updating order data
9	OrderRepository	This class handles data access for order records
10	IUnitOfWork	This interface provides SaveChangeAsync for transaction management
11	UnitOfWork	This class implements IUnitOfWork to commit changes to the database
12	IRepositoryBase	This is a generic interface for base repository operations
13	RepositoryBase	This class provides reusable update logic for repository classes
14	Order	This class represents a customer order with fields and methods such as SetOrderCode
15	ApplicationDbContext	This class manages the EF Core database context and database connection

Figure 106 - Create Payment QR Class Specification

3.11 <Staff> Create Payment Cash

3.11.1 Class Diagram

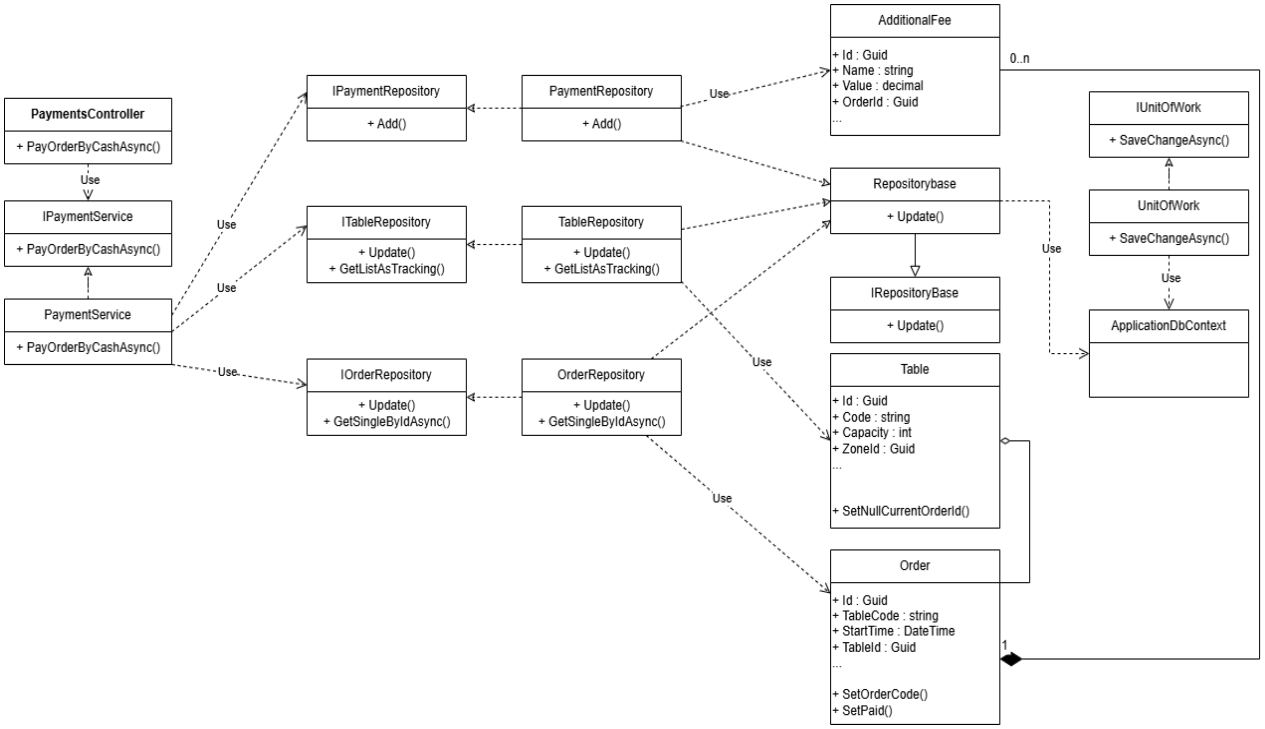


Figure 107 - Create Payment Cash class diagram

3.11.2 Sequence Diagram

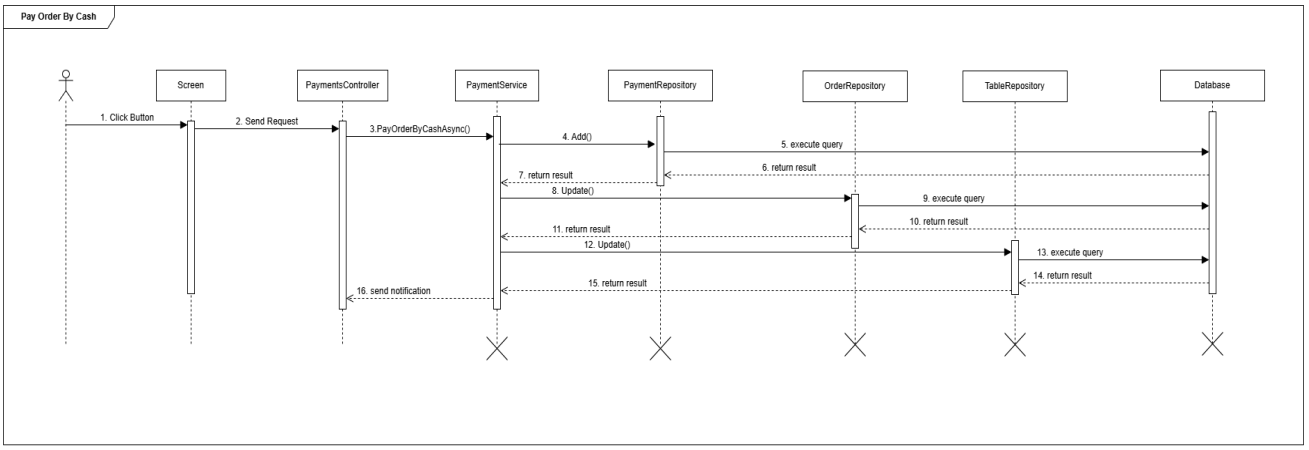


Figure 108 - Create Payment Cash sequence diagram

3.11.3 Class Specification

No	Class	Description
1	PaymentsController	This class handles HTTP requests related to payment operations

2	IPaymentService	This is an interface that defines the PayOrderByCashAsync method
3	PaymentService	This class implements logic for processing order payments by cash
4	IPaymentRepository	This is an interface for adding new payment records
5	PaymentRepository	This class handles data access for payment entities
6	ITableRepository	This is an interface for retrieving and updating table data
7	TableRepository	This class manages data access for table records
8	IOrderRepository	This is an interface for retrieving and updating order data
9	OrderRepository	This class handles data access for order entities
10	IUnitOfWork	This interface defines SaveChangeAsync for managing transactions
11	UnitOfWork	This class implements IUnitOfWork to persist changes in a transactional context
12	IRepositoryBase	This is a generic repository interface providing update functionality
13	RepositoryBase	This class implements base repository logic for data access
14	Table	This class represents a table in the system with code, capacity, and zone info
15	Order	This class represents a customer's order and includes methods like SetPaid
16	AdditionalFee	This class defines additional charges associated with an order
17	ApplicationDbContext	This class manages the Entity Framework Core database context

Figure 109 - Create Payment Cash Class Specification

3.12 <Manager> Create Workshop

3.12.1 Class Diagram

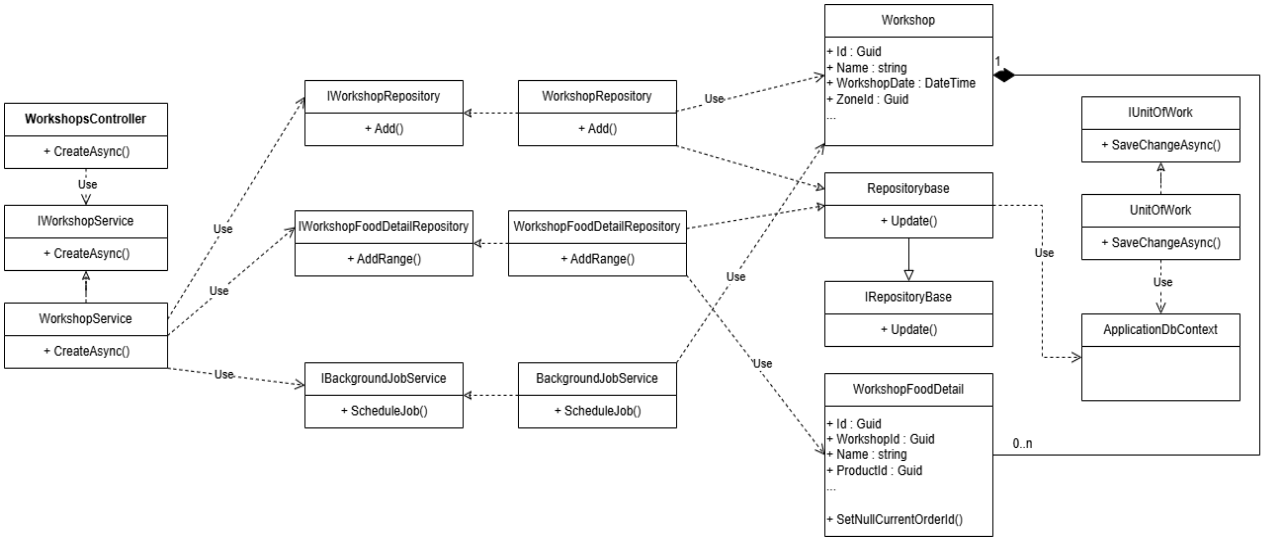


Figure 110 - Create Workshop class diagram

3.12.2 Sequence Diagram

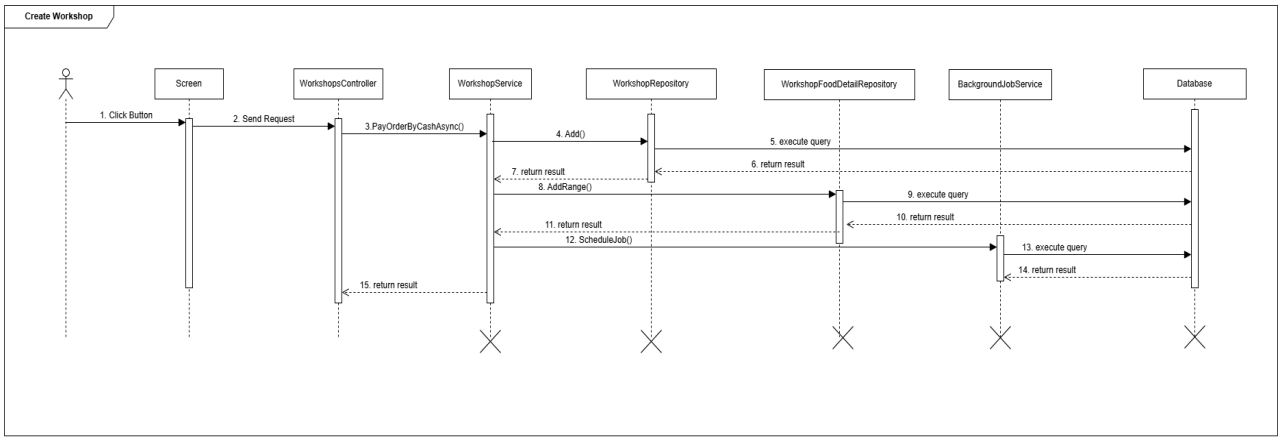


Figure 111- Create Workshop sequence diagram

3.12.3 Class Specification

No	Class	Description
1	WorkshopsController	This class handles HTTP requests for creating workshops
2	IWorkshopService	This is an interface that defines the CreateAsync method for workshop creation
3	WorkshopService	This class contains the business logic for creating workshops

4	IWorkshopRepository	This is an interface for adding workshop records
5	WorkshopRepository	This class handles data access for workshop entities
6	IWorkshopFoodDetailRepository	This is an interface for adding food detail records linked to a workshop
7	WorkshopFoodDetailRepository	This class manages data access for food details associated with a workshop
8	IBackgroundJobService	This is an interface for scheduling background jobs
9	BackgroundJobService	This class implements background job scheduling logic
10	IUnitOfWork	This interface defines SaveChangeAsync to manage atomic database operations
11	UnitOfWork	This class implements IUnitOfWork to commit changes transactionally
12	IRepositoryBase	This is a generic repository interface offering update support
13	RepositoryBase	This class provides common repository update logic
14	Workshop	This class represents a workshop session with name, date, and zone info
15	WorkshopFoodDetail	This class defines a link between a workshop and a product used in that session
16	ApplicationDbContext	This class manages the Entity Framework Core database context

Figure 112 - Create Workshop Class Specification

3.13 <Customer> Register Workshop

3.13.1 Class Diagram

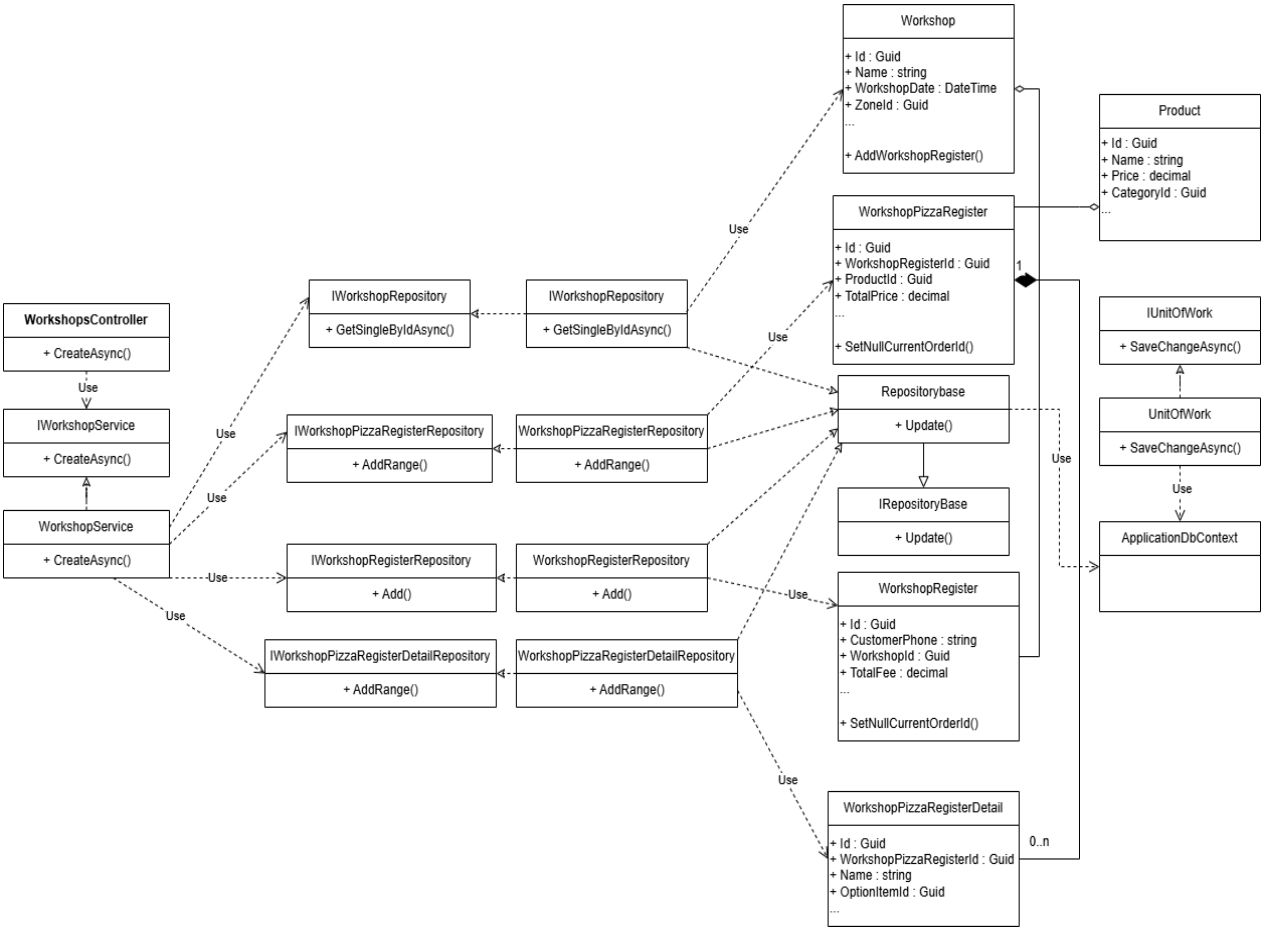


Figure 113 - Register Workshop class diagram

3.13.2 Sequence Diagram

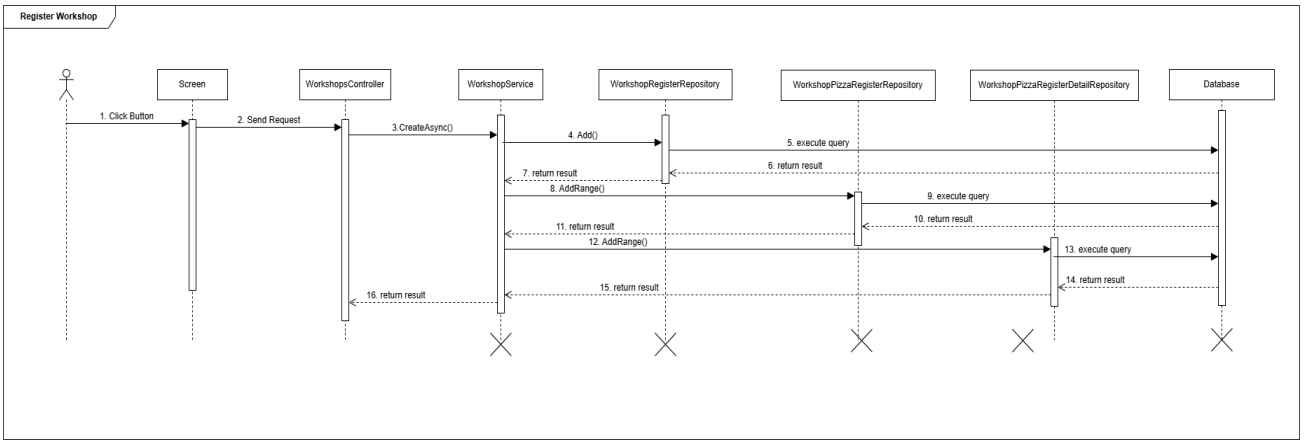


Figure 114- Register Workshop sequence diagram

3.13.3 Class Specification

No	Class	Description
1	WorkshopsController	This class handles HTTP requests for creating workshop sessions
2	IWorkshopService	This is an interface defining the method to create a workshop session
3	WorkshopService	This class implements the logic for creating workshop sessions
4	IWorkshopRepository	This is an interface for retrieving workshop data
5	WorkshopRepository	This class handles data access for workshop entities
6	IWorkshopRegisterRepository	This is an interface for managing workshop registration data
7	WorkshopRegisterRepository	This class handles data access for workshop registration records
8	IWorkshopPizzaRegisterRepository	This is an interface for adding workshop-pizza registration records
9	WorkshopPizzaRegisterRepository	This class handles data access for workshop-pizza registration entities
10	IWorkshopPizzaRegisterDetailRepository	This is an interface for managing pizza register details in workshops
11	WorkshopPizzaRegisterDetailRepository	This class handles data access for pizza register detail records
12	IUnitOfWork	This interface defines SaveChangeAsync for transactional operations
13	UnitOfWork	This class implements IUnitOfWork to commit changes to the database
14	IRepositoryBase	This is a generic repository interface for update operations
15	RepositoryBase	This class provides reusable base repository functionality
16	Workshop	This class represents a workshop event with name, date, and zone info
17	Product	This class defines a product used in workshops with name, price, and category
18	WorkshopPizzaRegister	This class represents the registration of a pizza product in a workshop
19	WorkshopRegister	This class represents a customer registration for a workshop session
20	WorkshopPizzaRegisterDetail	This class holds extra detail information linked to a pizza registration
21	ApplicationDbContext	This class manages the database context and entity configurations

Figure 115- Register Workshop Class Specification

3.14 <Customer> Check In Workshop

3.14.1 Class Diagram

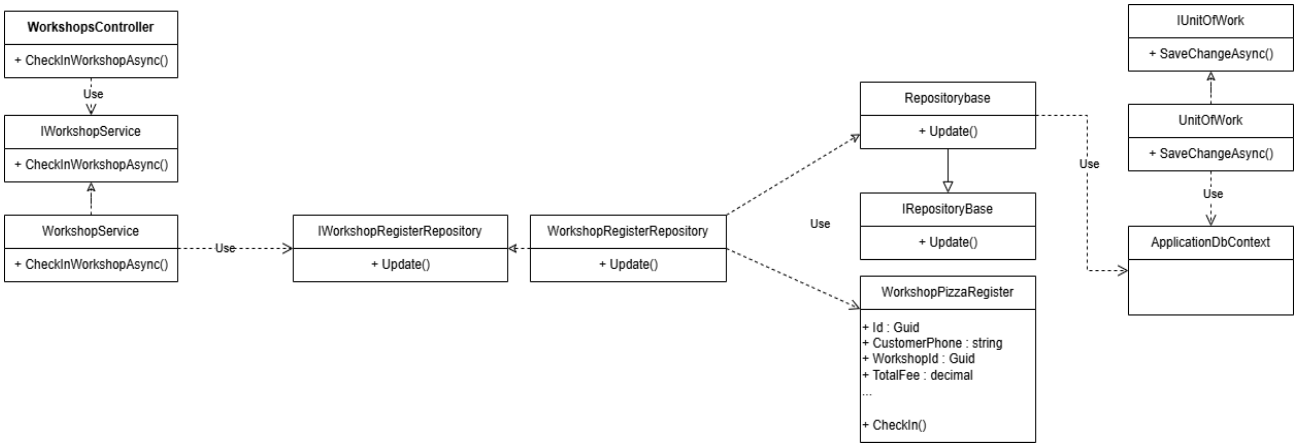


Figure 116 - Check In Workshop class diagram

3.14.2 Sequence Diagram

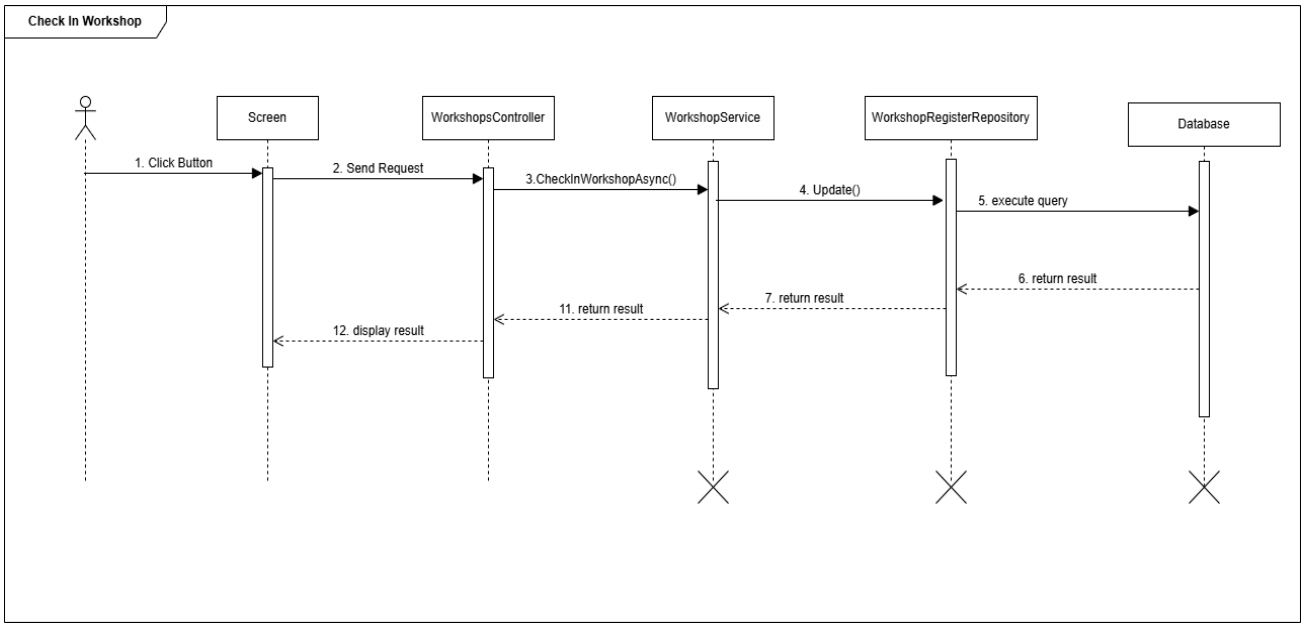


Figure 117 - Check In Workshop sequence diagram

3.14.3 Class Specification

No	Class	Description
1	WorkshopsController	This class handles HTTP requests related to assigning workshop registrations
2	IWorkshopService	This is an interface defining the method to assign workshop registrations
3	WorkshopService	This class contains the logic for assigning workshop

		registrations
4	ITableRepository	This is an interface for retrieving and updating table records
5	TableRepository	This class handles data access for table entities
6	IWorkshopRegisterRepository	This is an interface for accessing and updating workshop register data
7	WorkshopRegisterRepository	This class manages data access for workshop register records
8	IOrderItemRepository	This is an interface for bulk adding order item records
9	OrderItemRepository	This class handles data access for order item entities
10	IOrderRepository	This is an interface for adding order records
11	OrderRepository	This class handles data access for order entities
12	IUnitOfWork	This interface defines SaveChangeAsync for handling transactional operations
13	UnitOfWork	This class implements IUnitOfWork for committing changes
14	IRepositoryBase	This is a generic interface providing update functionality
15	RepositoryBase	This class provides base repository functionality shared across repositories
16	Table	This class represents a restaurant table, including code, capacity, and zone assignment
17	WorkshopPizzaRegister	This class represents a registered product in a workshop session
18	Order	This class represents a customer's order including table and time data
19	OrderItem	This class represents an item in an order including name, quantity, and price
20	ApplicationDbContext	This class manages the EF Core database context and connections

Figure 118 - Check In Workshop Class Specification

3.15 <Staff> Assign Table for Workshop Register

3.15.1 Class Diagram

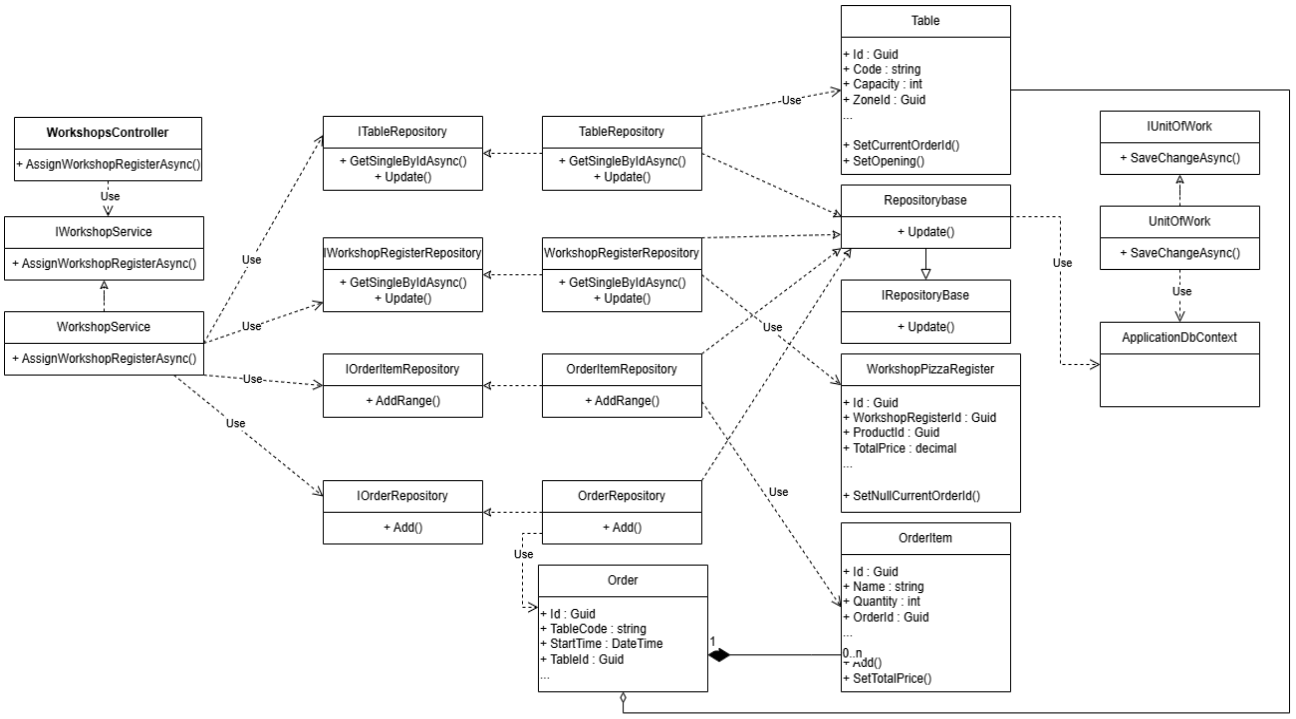


Figure 119 - Assign Table for Workshop Register class diagram

3.15.2 Sequence Diagram

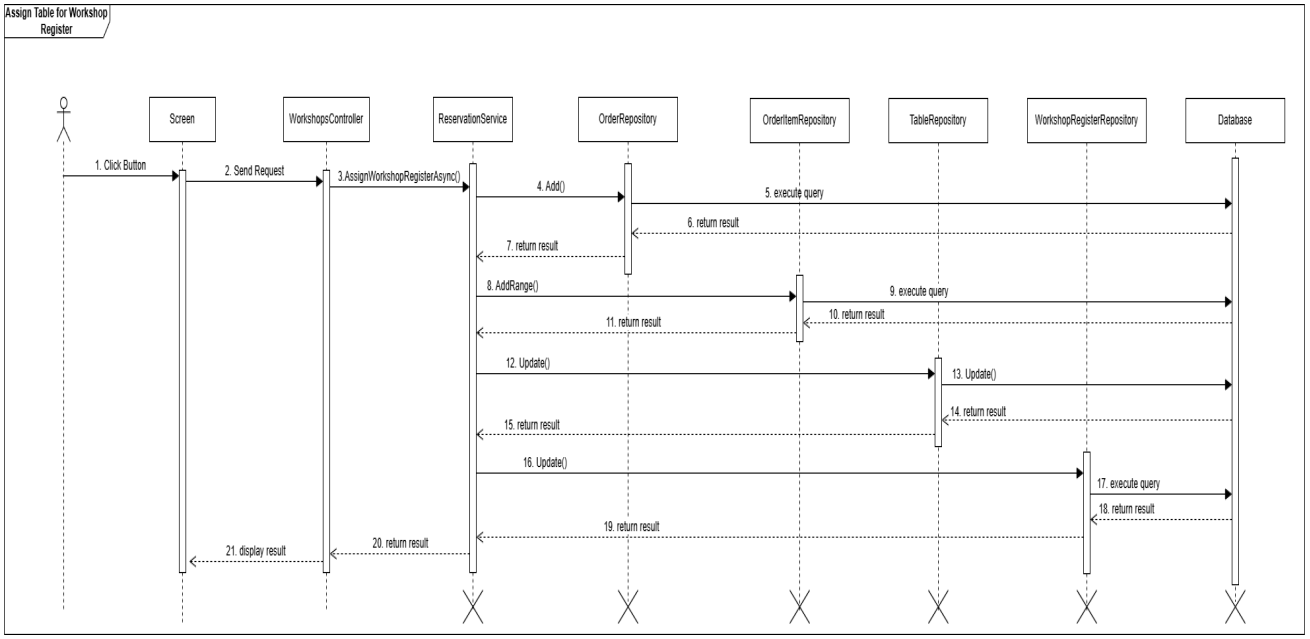


Figure 120- Assign Table for Workshop Register sequence diagram

3.15.3 Class Specification

No	Class	Description
1	WorkshopsController	This class handles HTTP requests related to assigning workshop registrations
2	IWorkshopService	This is an interface defining the method to assign workshop registrations
3	WorkshopService	This class contains the logic for assigning workshop registrations
4	ITableRepository	This is an interface for retrieving and updating table records
5	TableRepository	This class handles data access for table entities
6	IWorkshopRegisterRepository	This is an interface for accessing and updating workshop register data
7	WorkshopRegisterRepository	This class manages data access for workshop register records
8	IOrderItemRepository	This is an interface for bulk adding order item records
9	OrderItemRepository	This class handles data access for order item entities
10	IOrderRepository	This is an interface for adding order records
11	OrderRepository	This class handles data access for order entities
12	IUnitOfWork	This interface defines SaveChangeAsync for handling transactional operations
13	UnitOfWork	This class implements IUnitOfWork for committing changes
14	IRepositoryBase	This is a generic interface providing update functionality
15	RepositoryBase	This class provides base repository functionality shared across repositories
16	Table	This class represents a restaurant table, including code, capacity, and zone assignment
17	WorkshopPizzaRegister	This class represents a registered product in a workshop session
18	Order	This class represents a customer's order including table and time data
19	OrderItem	This class represents an item in an order including name, quantity, and price
20	ApplicationDbContext	This class manages the EF Core database context and connections

Figure 121- Assign Table for Workshop Register Class Specification

3.16 <Staff> Register Working Slot

3.16.1 Class Diagram

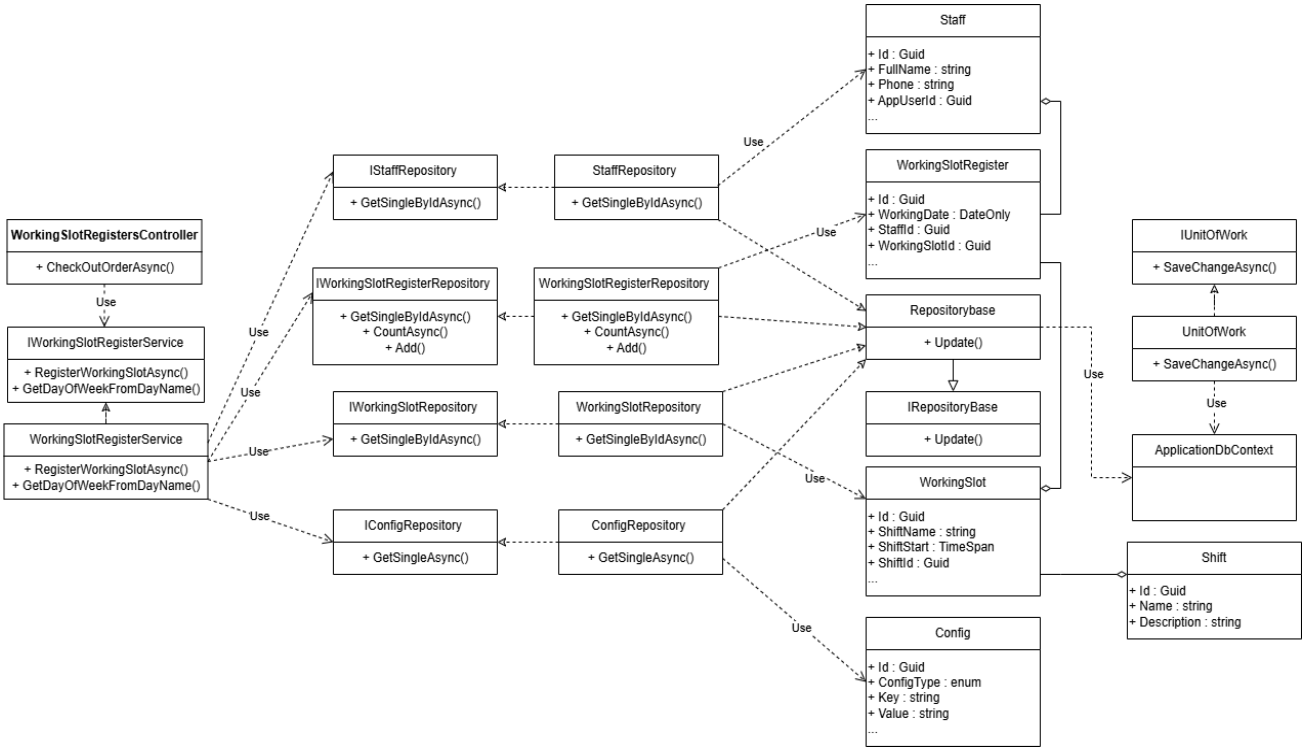


Figure 122- Register Working Slot class diagram

3.16.2 Sequence Diagram

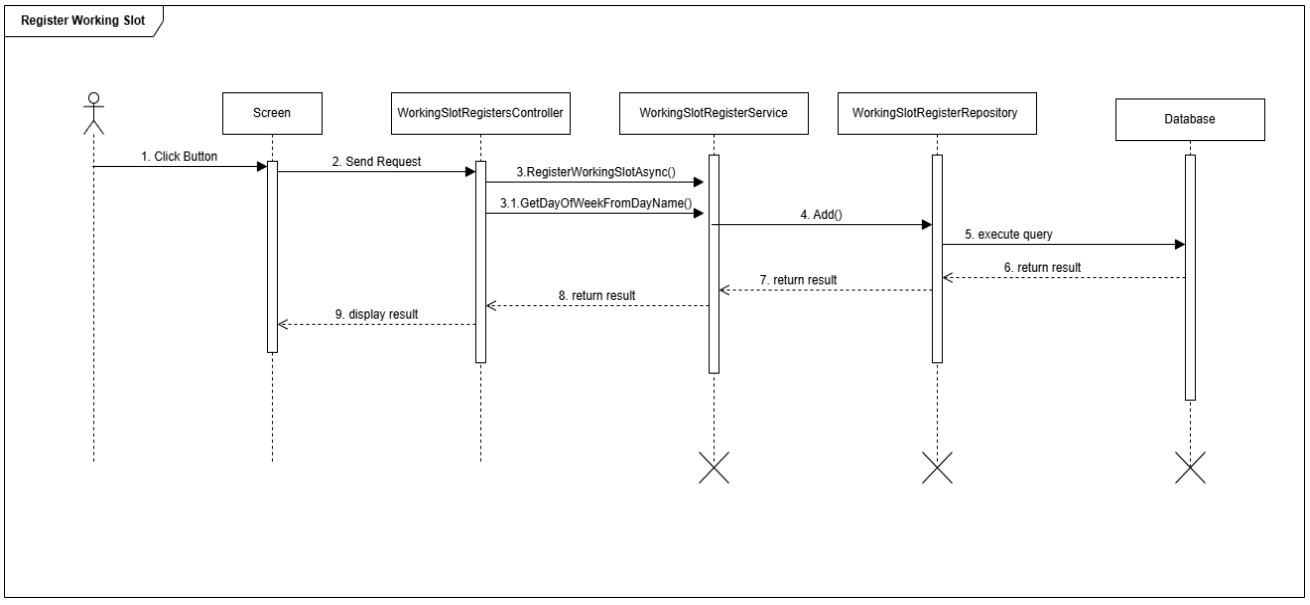


Figure 123- Register Working Slot sequence diagram

3.16.3 Class Specification

No	Class	Description
1	WorkingSlotRegistersController	This class handles HTTP requests related to working slot registration
2	IWorkingSlotRegisterService	This is an interface defining methods for registering and querying working slot data
3	WorkingSlotRegisterService	This class implements logic for registering working slots and handling related operations
4	IStaffRepository	This is an interface for retrieving staff-related data
5	StaffRepository	This class handles data access for staff entities
6	IWorkingSlotRegisterRepository	This is an interface for managing working slot register records, including add and count operations
7	WorkingSlotRegisterRepository	This class handles data access for working slot register entities
8	IWorkingSlotRepository	This is an interface for retrieving working slot data
9	WorkingSlotRepository	This class handles data access for working slot entities
10	IConfigRepository	This is an interface for accessing configuration records
11	ConfigRepository	This class handles data access for config entities
12	IUnitOfWork	This interface defines SaveChangeAsync for transactional control
13	UnitOfWork	This class implements IUnitOfWork to persist changes as a unit of work
14	IRepositoryBase	This is a generic repository interface for common operations like update
15	RepositoryBase	This class implements base repository functionality
16	Staff	This class represents a staff member with identity and contact details
17	WorkingSlotRegister	This class represents a staff's registration to a working slot on a specific date
18	WorkingSlot	This class defines a working slot, including shift name and start time
19	Config	This class contains configuration settings identified by key-value pairs and config type
20	ApplicationDbContext	This class manages the database context and bridges entity models to the database
21	Shift	This class defines a shift entity with a name and description

Figure 124- Register Working Slot Class Specification

3.17 <Manager> Approve Working Slot Registration

3.17.1 Class Diagram

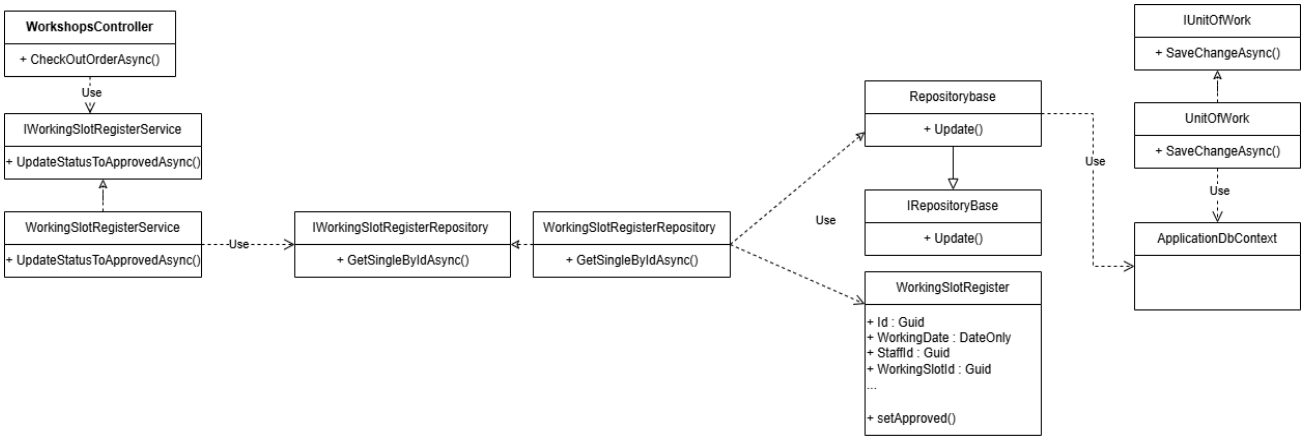


Figure 125- Approve Working Slot Registration class diagram

3.17.2 Sequence Diagram

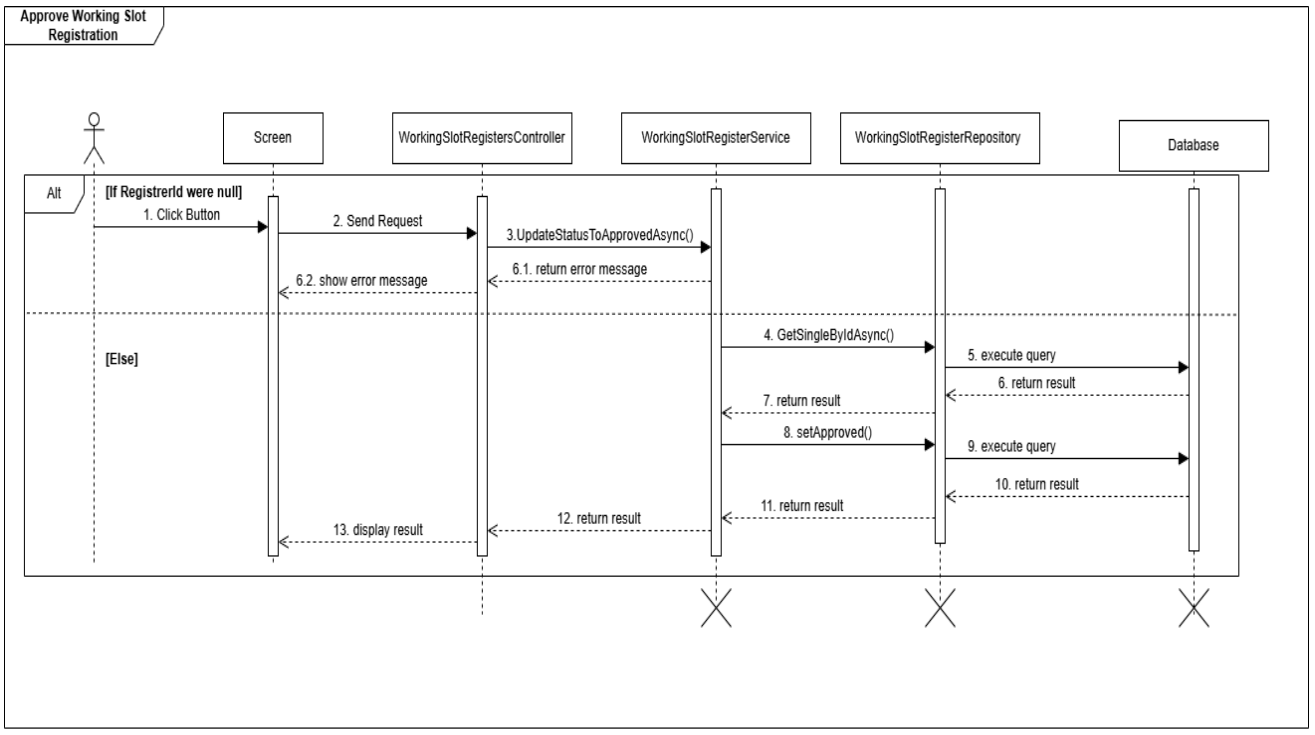


Figure 126- Approve Working Slot Registration sequence diagram

3.17.3 Class Specification

No	Class	Description
1	WorkshopsController	This class handles HTTP requests related to workshop operations

2	IWorkingSlotRegisterService	This is an interface defining the method to update working slot register status
3	WorkingSlotRegisterService	This class contains business logic for updating the status of working slot registers
4	IWorkingSlotRegisterRepository	This is an interface for accessing specific working slot register records
5	WorkingSlotRegisterRepository	This class handles data access for working slot registration records
6	IUnitOfWork	This interface defines a method to commit changes via SaveChangeAsync
7	UnitOfWork	This class implements transactional commits for changes made in the service layer
8	IRepositoryBase	This is a generic repository interface providing common data update operations
9	RepositoryBase	This class implements common repository functionality shared across repositories
10	WorkingSlotRegister	This class represents a working slot registration for a staff on a given date
11	ApplicationDbContext	This class manages the database context and acts as a bridge to EF Core

Figure 127- Approve Working Slot Registration Class Specification

3.18 <Manager> Assign Staff to Zone Schedule

3.18.1 Class Diagram

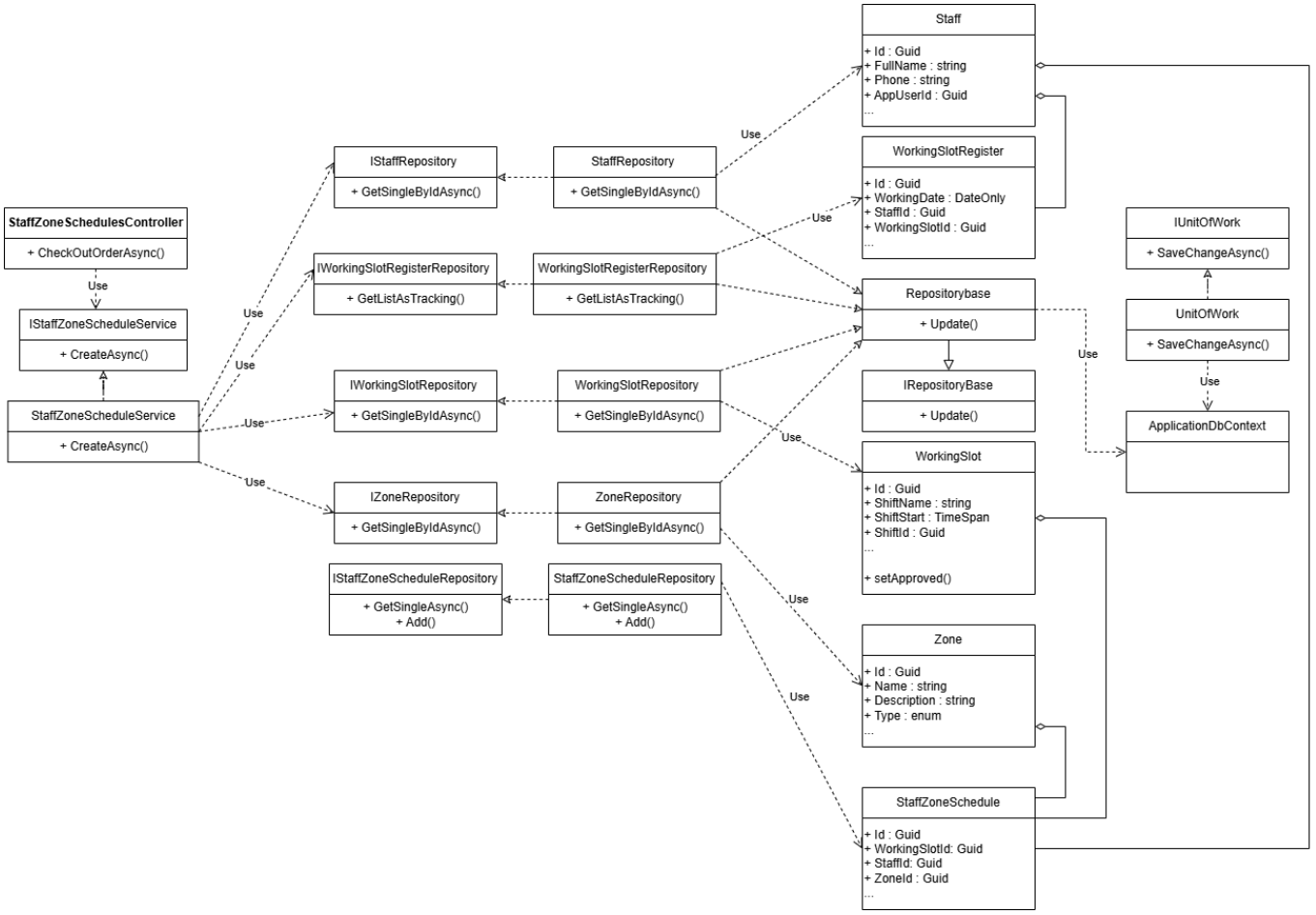


Figure 128- Assign Staff to Zone Schedule class diagram

3.18.2 Sequence Diagram

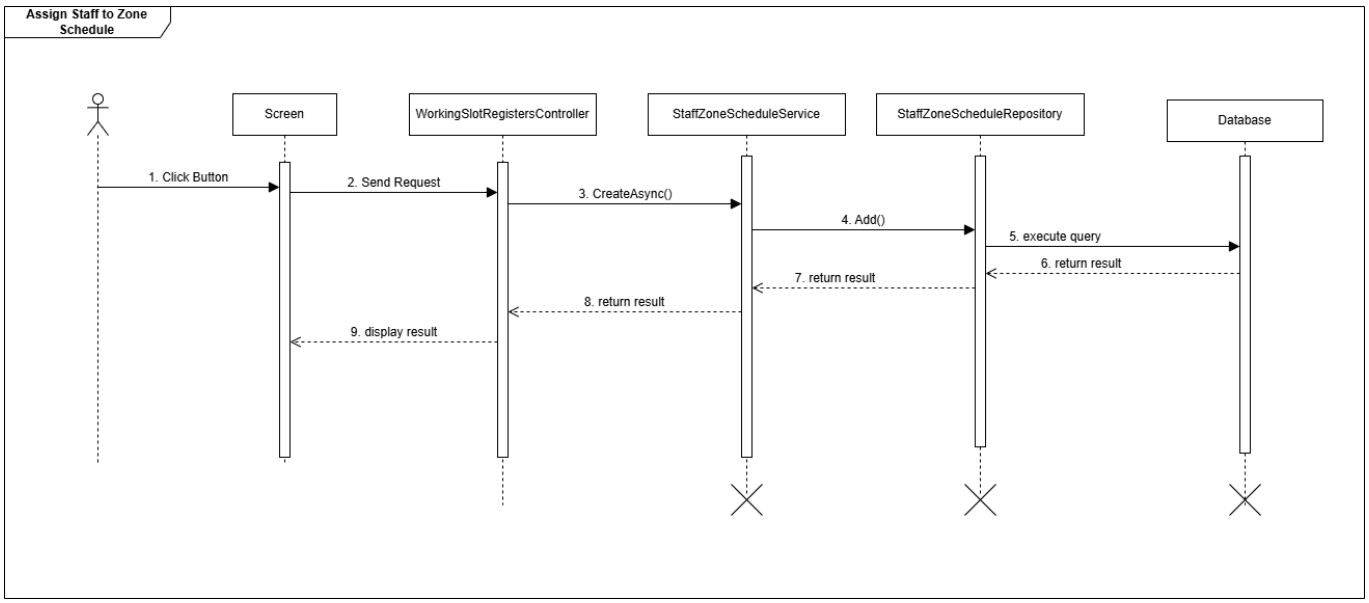


Figure 129- Assign Staff to Zone Schedule sequence diagram

3.18.3 Class Specification

No	Class	Description
1	StaffZoneSchedulesController	This class handles HTTP requests for staff zone scheduling actions
2	IStaffZoneScheduleService	This is an interface that defines methods implemented by StaffZoneScheduleService
3	StaffZoneScheduleService	This class contains business logic for creating staff zone schedules
4	IStaffRepository	This is an interface for retrieving staff data
5	StaffRepository	This class handles data access for staff entities
6	IWorkingSlotRegisterRepository	This is an interface for accessing working slot registration records
7	WorkingSlotRegisterRepository	This class handles data access for working slot registration entities
8	IWorkingSlotRepository	This is an interface for retrieving working slot data
9	WorkingSlotRepository	This class handles data access for working slot entities
10	IZoneRepository	This is an interface for accessing zone data
11	ZoneRepository	This class handles data access for zone entities
12	IStaffZoneScheduleRepository	This is an interface for managing staff zone schedule data
13	StaffZoneScheduleRepository	This class handles data access for staff zone schedule entities
14	IUnitOfWork	This interface defines SaveChangeAsync for handling transactional operations

15	UnitOfWork	This class implements IUnitOfWork to commit database changes
16	IRepositoryBase	This is a generic repository interface with update method support
17	RepositoryBase	This class provides base repository functionality reused across repositories
18	Staff	This class represents a staff entity with properties like name, phone, and AppUserId
19	WorkingSlotRegister	This class represents staff registration to a working slot on a specific date
20	WorkingSlot	This class represents a working slot with shift name and timing info
21	Zone	This class defines a zone with a name, description, and type
22	StaffZoneSchedule	This class maps a staff member to a specific zone in a working slot
23	ApplicationDbContext	This class handles the database context and configuration setup

Figure 130- Assign Staff to Zone Schedule Class Specification

3.19 <Manager> Auto Assign Staff to Zone Schedule

3.19.1 Class Diagram

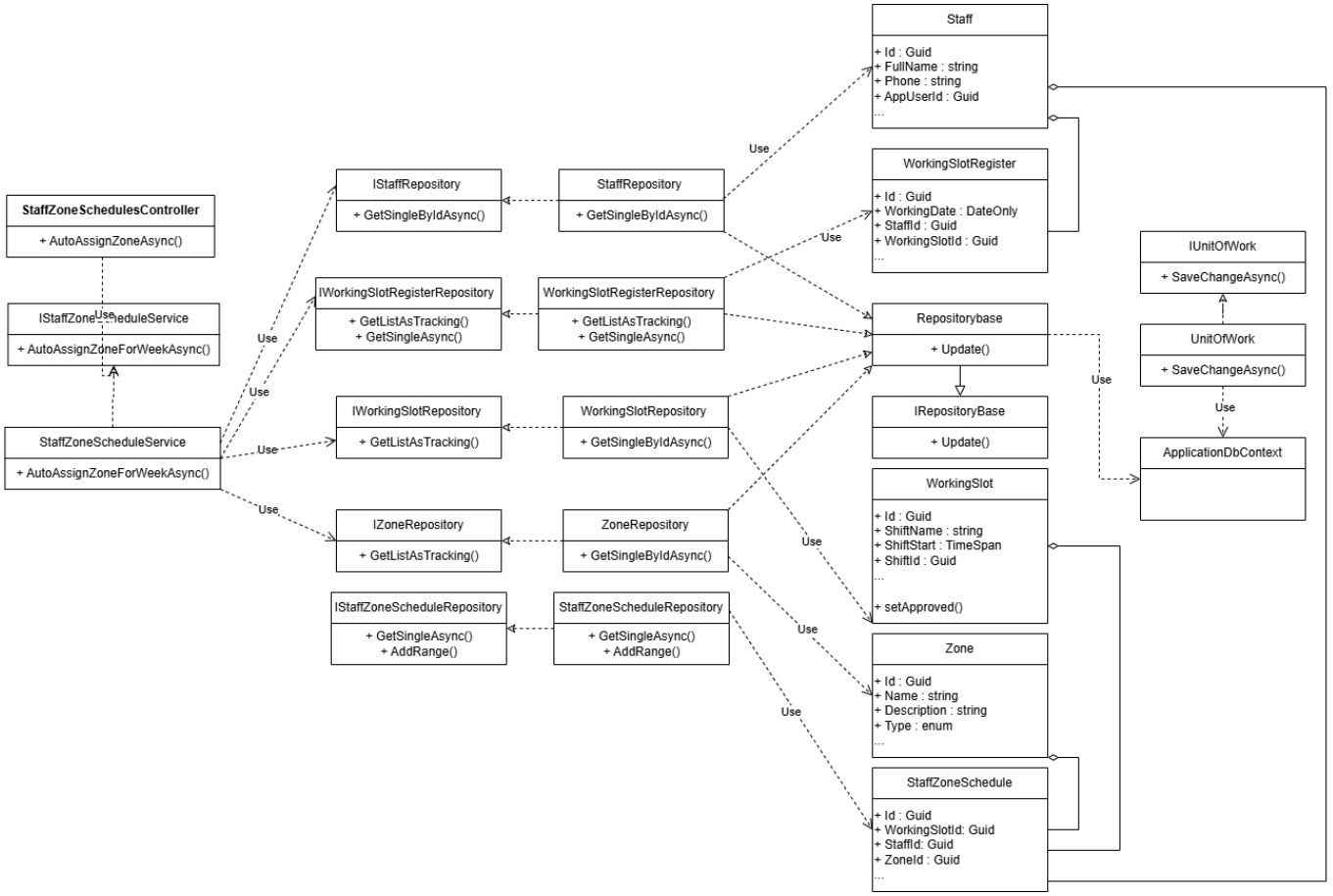


Figure 131- Auto Assign Staff to Zone Schedule class diagram

3.19.2 Sequence Diagram

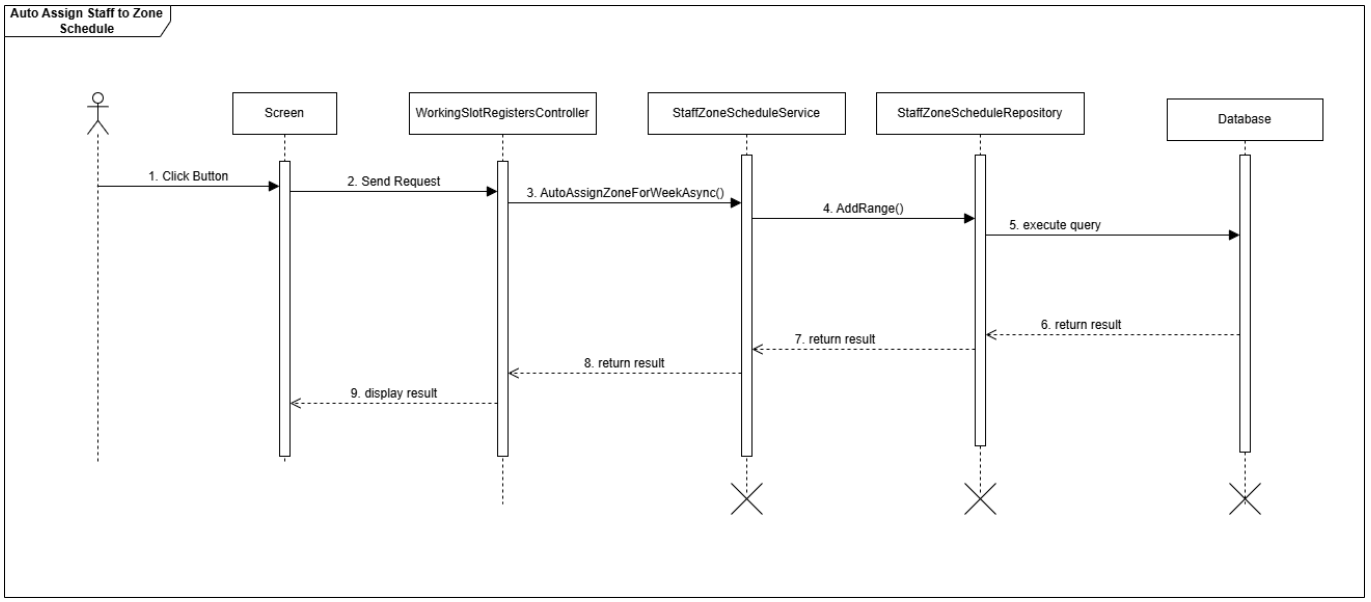


Figure 132- Auto Assign Staff to Zone Schedule sequence diagram

3.19.3 Class Specification

No	Class	Description
1	StaffZoneSchedulesController	This class handles HTTP requests related to automatic staff-zone scheduling
2	IStaffZoneScheduleService	This is an interface that defines methods for scheduling staff to zones
3	StaffZoneScheduleService	This class contains business logic for auto-assigning zones to staff
4	IStaffRepository	This is an interface for retrieving staff data
5	StaffRepository	This class handles data access for staff
6	IWorkingSlotRegisterRepository	This is an interface for managing working slot register data
7	WorkingSlotRegisterRepository	This class handles data access for working slot registration records
8	IWorkingSlotRepository	This is an interface for accessing working slot data
9	WorkingSlotRepository	This class handles data access for working slot entities
10	IZoneRepository	This is an interface for retrieving zone data
11	ZoneRepository	This class handles data access for zone entities
12	IStaffZoneScheduleRepository	This is an interface for managing staff-zone schedule data
13	StaffZoneScheduleRepository	This class handles data access for staff-zone scheduling
14	IUnitOfWork	This interface defines a method to commit changes in a unit-of-work pattern
15	UnitOfWork	This class implements SaveChangeAsync to persist changes transactionally
16	IRepositoryBase	This is a generic repository interface providing base CRUD operations
17	RepositoryBase	This class provides base repository logic shared across repositories
18	Staff	This class represents a staff entity with personal info
19	WorkingSlotRegister	This class represents registration of a staff member to a working slot
20	WorkingSlot	This class defines a shift-based working slot with time information
21	Zone	This class represents a service area/zone assigned to staff
22	StaffZoneSchedule	This class maps staff to zones during specific working slots
23	ApplicationDbContext	This class manages database configuration and access

Figure 133- Auto Assign Staff to Zone Schedule Class Specification

3.20 <Staff> Swap Working Slot

3.20.1 Class Diagram

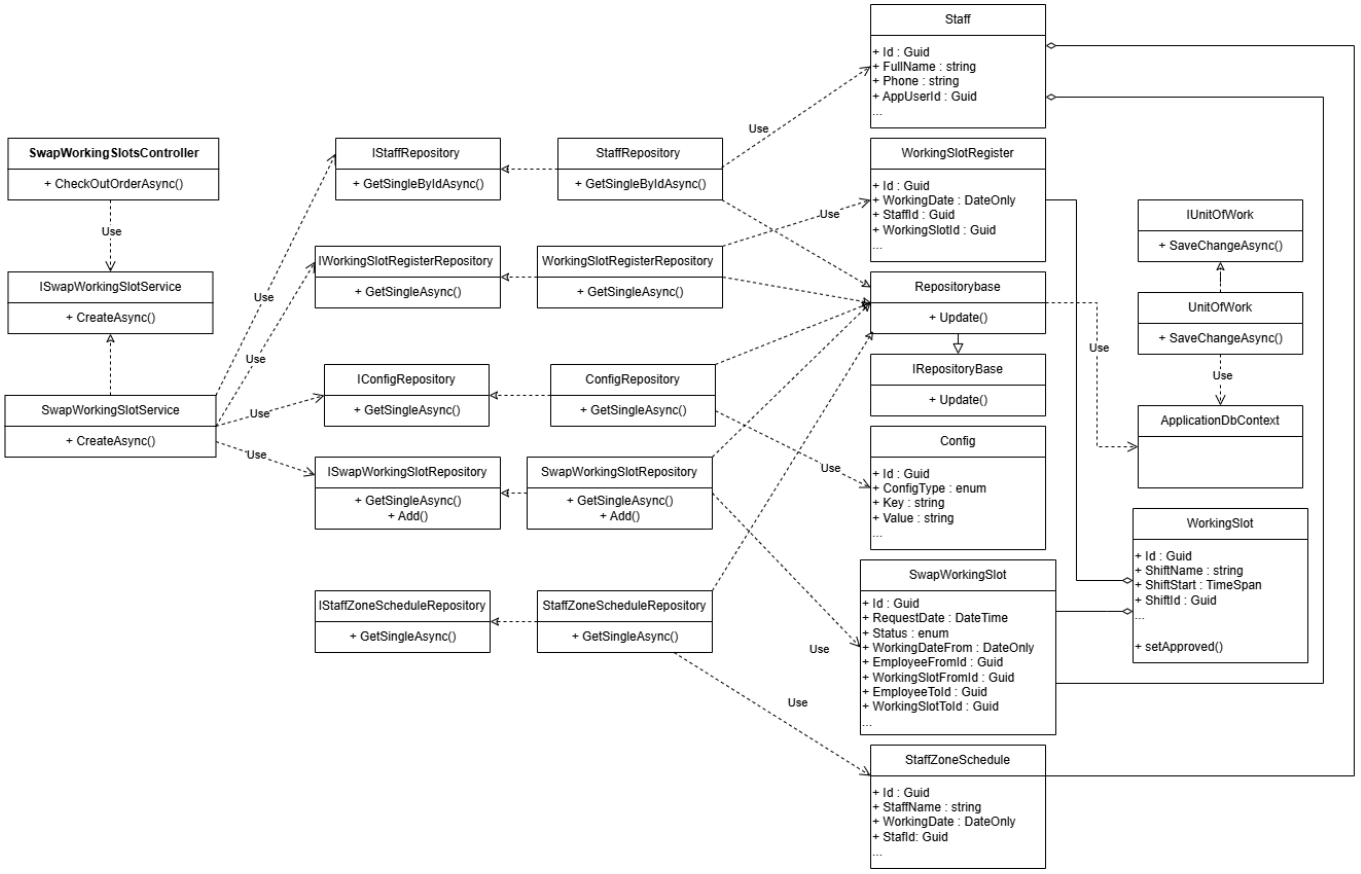


Figure 134- Swap Working Slot class diagram

3.20.2 Sequence Diagram

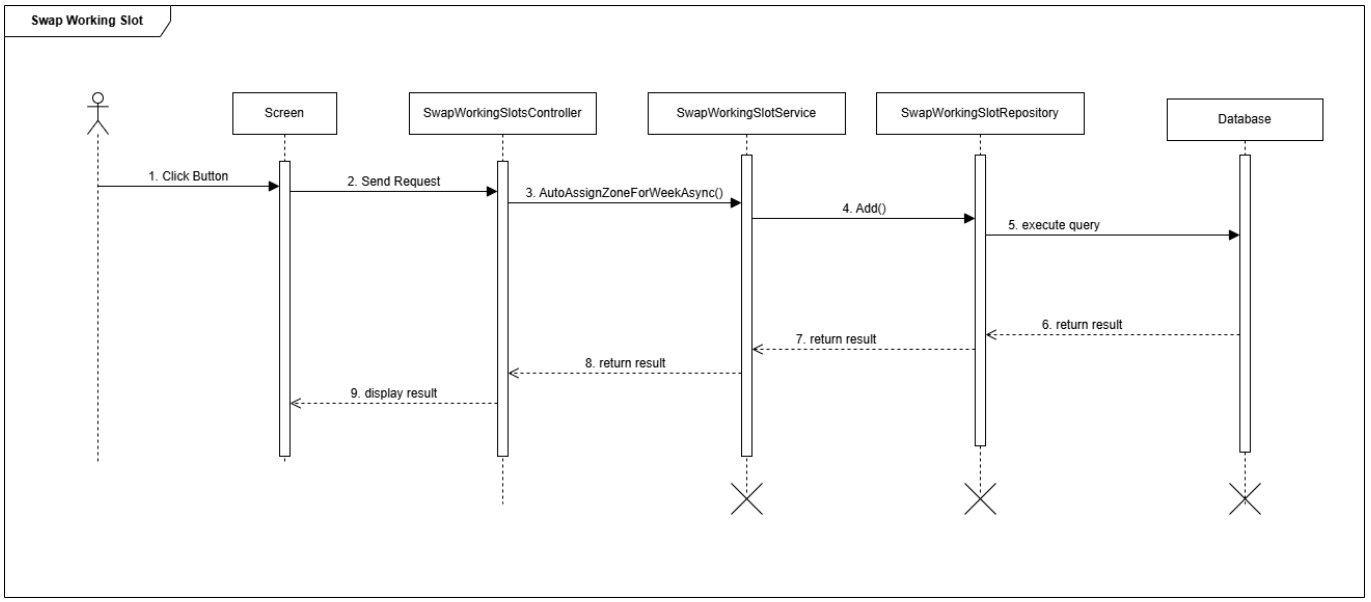


Figure 135- Swap Working Slot sequence diagram

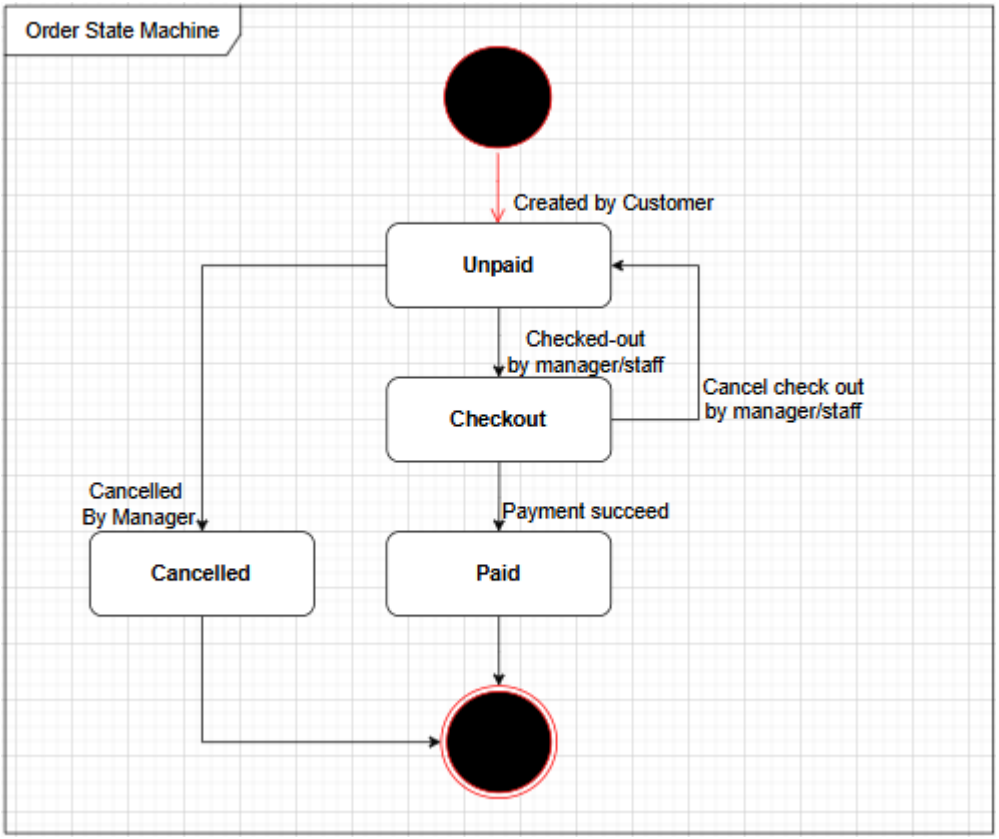
3.20.3 Class Specification

1	SwapWorkingSlotsController	This class handles HTTP requests related to working slot swapping
2	ISwapWorkingSlotService	This is an interface that defines methods implemented by SwapWorkingSlotService
3	SwapWorkingSlotService	This class contains business logic for processing swap working slot requests
4	IStaffRepository	This is an interface for retrieving staff data
5	StaffRepository	This class handles data access for staff
6	IWorkingSlotRegisterRepository	This is an interface for retrieving working slot register data
7	WorkingSlotRegisterRepository	This class handles data access for working slot registers
8	IConfigRepository	This is an interface for accessing configuration settings
9	ConfigRepository	This class handles data access for configuration entities
10	ISwapWorkingSlotRepository	This is an interface for accessing swap working slot records
11	SwapWorkingSlotRepository	This class handles data access for swap working slot entities
12	IStaffZoneScheduleRepository	This is an interface for accessing staff zone schedules
13	StaffZoneScheduleRepository	This class handles data access for staff zone schedule entities
14	IUnitOfWork	This is an interface defining the unit of work pattern with SaveChangeAsync
15	UnitOfWork	This class implements transaction control via SaveChangeAsync
16	IRepositoryBase	This is a generic repository interface providing update methods
17	RepositoryBase	This class provides base repository logic shared among repositories
18	Staff	This class represents a staff entity with personal and user info
19	WorkingSlotRegister	This class represents a record of a staff member registered to a slot
20	Config	This class stores configuration key-value pairs
21	SwapWorkingSlot	This class represents a request to swap working slots between employees
22	StaffZoneSchedule	This class represents a staff zone assignment schedule
23	WorkingSlot	This class defines a working slot with shift information
24	ApplicationDbContext	This class handles database context and configuration

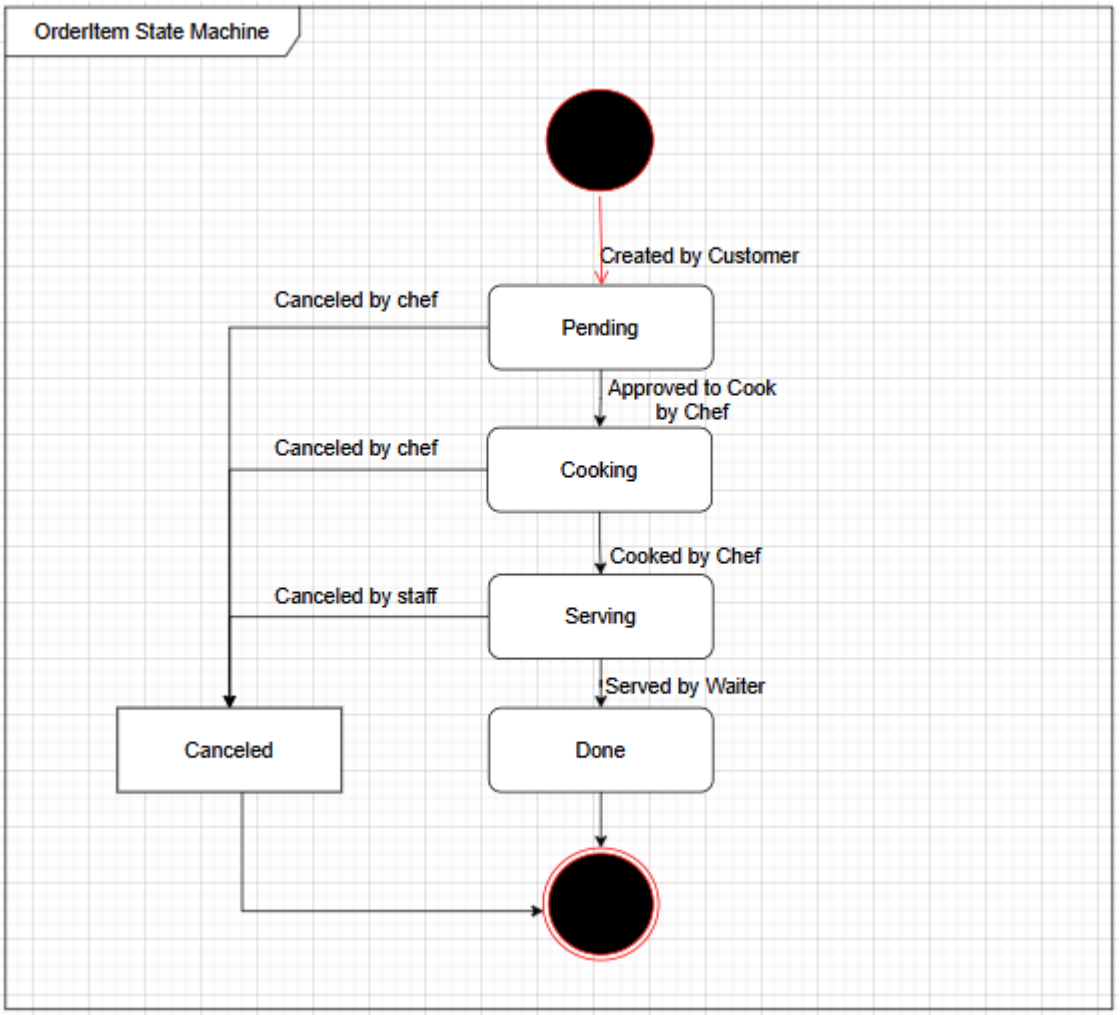
Figure 136- Swap Working Slot Class Specification

4. State machine diagram

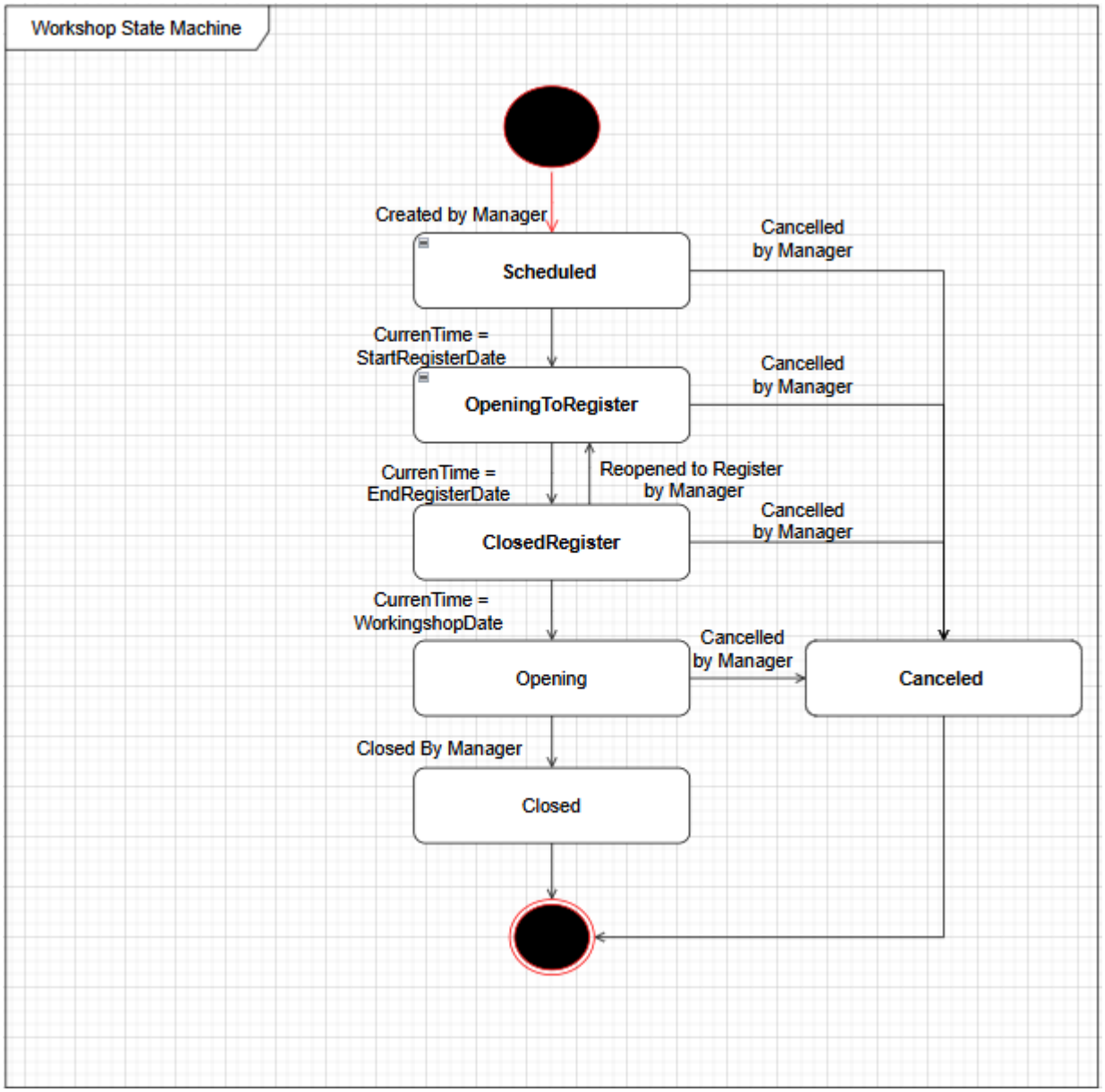
4.1 Order State Machine Diagram



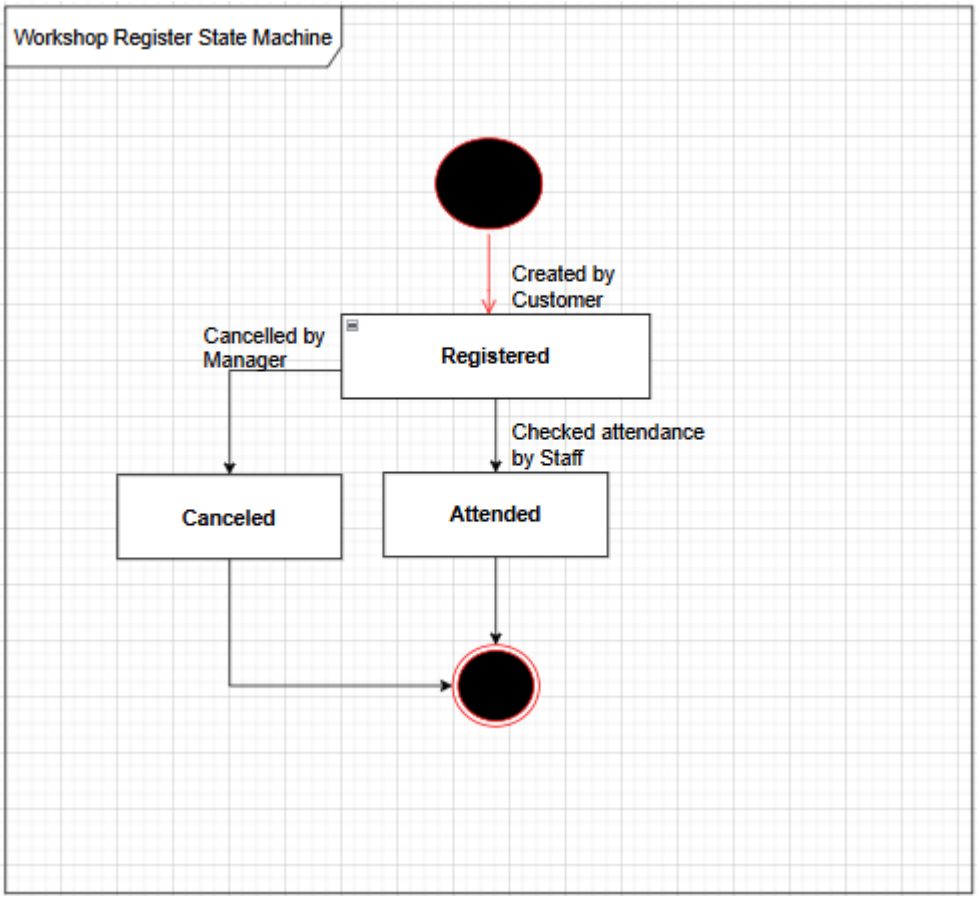
4.2 Order Item State Machine Diagram



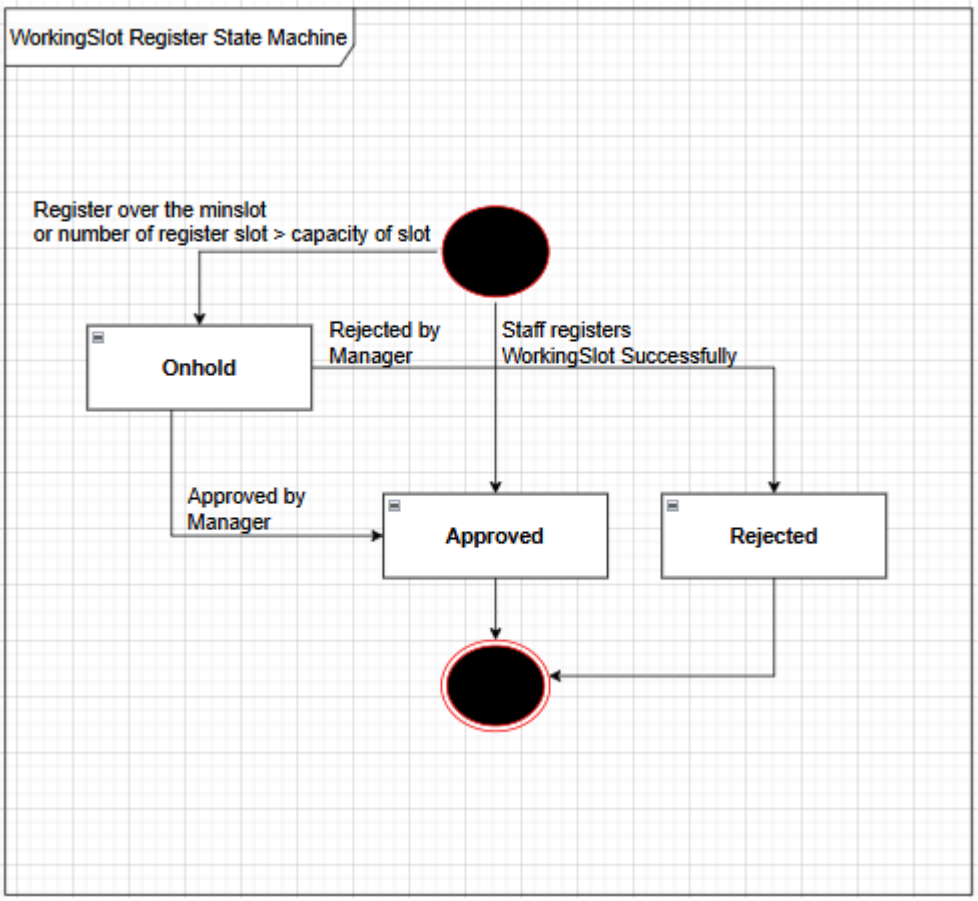
4.3 Workshop State Machine Diagram



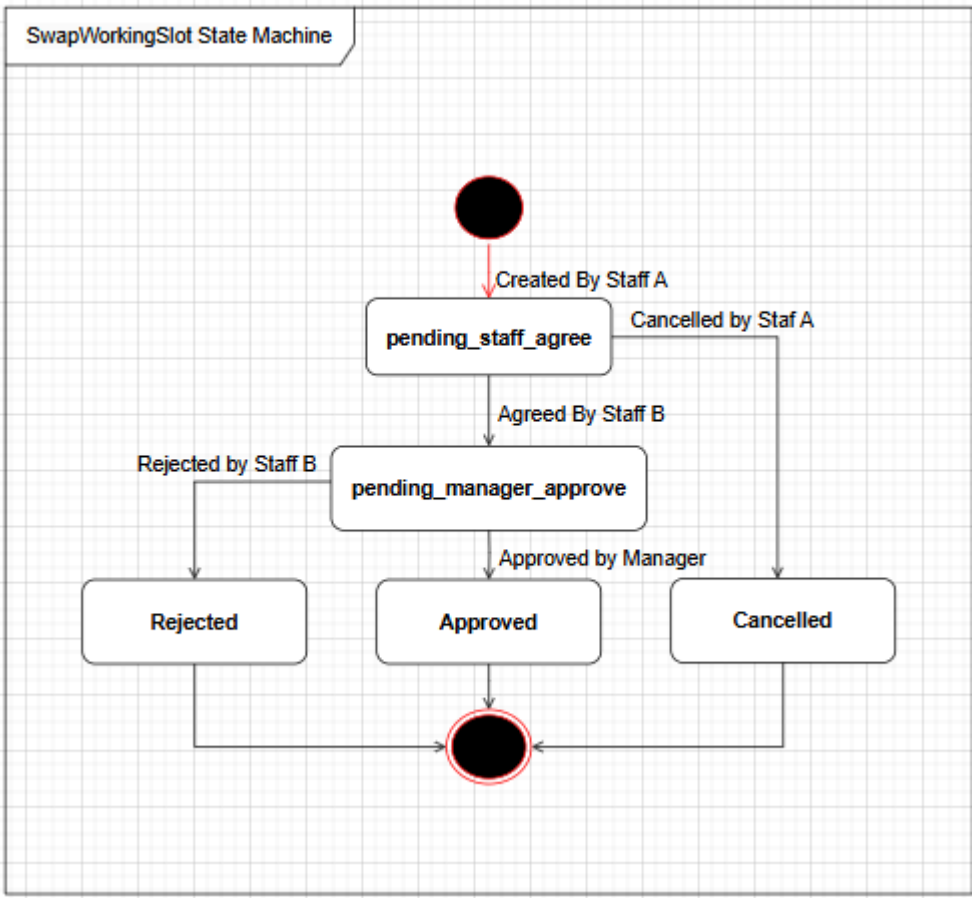
4.4 Workshop Register State Machine Diagram



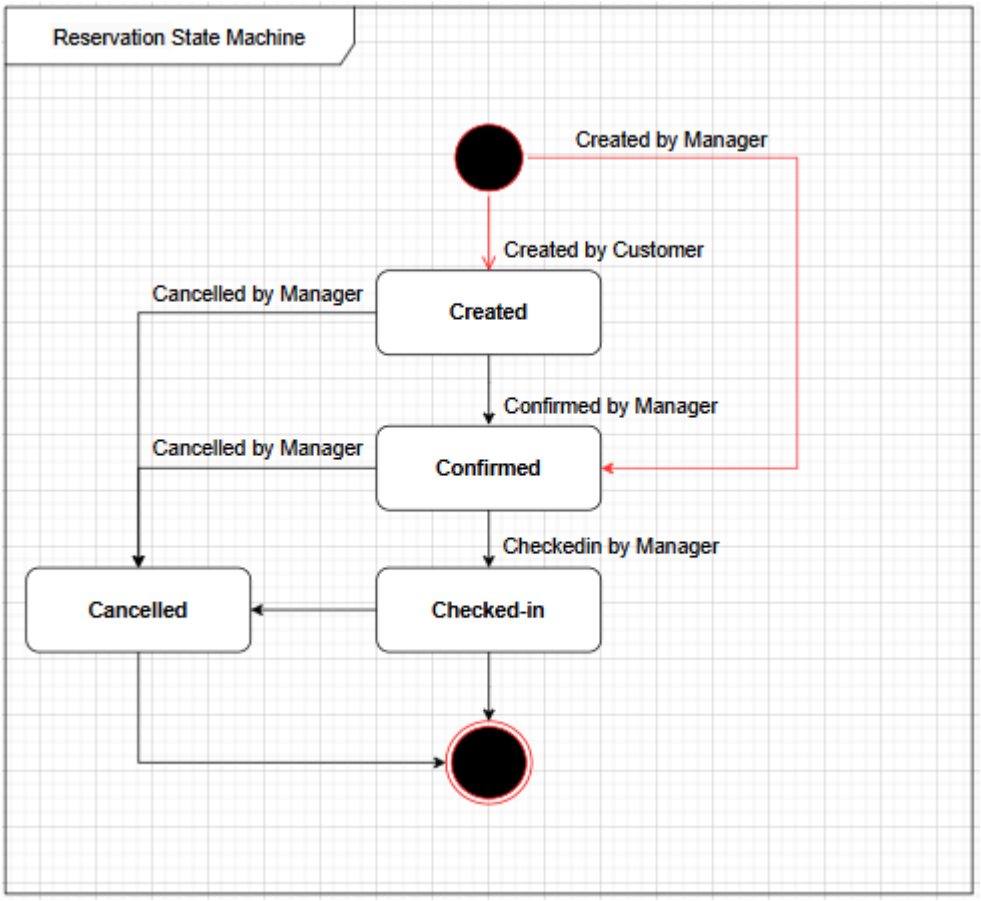
4.5 Working Slot Register State Machine Diagram



4.6 Swap Working Slot Register State Machine Diagram



4.7 Reservation State Machine Diagram



V. Software Testing Documentation

1. Scope of Testing

1.1 Test Model

This project is developed using the Scrum model – part of an Agile framework for Software development projects. Scrum is suitable for small and medium projects.

We apply scrum testing for owner projects divided into 4 levels of testing: Unit testing, integration testing, system testing, and acceptance testing.

1.2 Testing Levels

1.2.1. Unit Testing

Software testing is an essential part of the software development life cycle, ensuring that the final product meets the required standards for functionality, performance, and reliability. The testing process is structured into several levels, each serving a specific purpose and performed by different roles within the project team.

1.2.2. Integration Testing

Following unit testing, integration testing focuses on verifying the interactions between integrated units. It is defined as the phase where individual modules are combined and tested as a group to identify interface defects between them. The objective is to expose faults that may arise when units are brought together, which may not have been apparent during unit testing. Integration testing is generally carried out by dedicated testers, although developers may also participate depending on the project practices.

There are three main approaches to integration testing:

- Top-down approach, where higher-level modules are tested first and lower-level modules are integrated sequentially.
- Bottom-up approach, where lower-level modules are tested and then integrated upward.
- Sandwich (or Hybrid) approach, which combines top-down and bottom-up techniques.

In our system, a Top-down integration testing approach is applied, ensuring that major control structures are tested early and subsidiary modules are progressively integrated and validated.

1.2.3. System Testing

System testing represents the next level, where the complete, integrated application is tested as a whole. This level evaluates the system's compliance with both functional and non-functional requirements, ensuring the software meets specified business and technical standards. System testing examines overall application behavior, system performance under load, usability, security, and other critical quality attributes. It verifies that the system functions correctly in its intended environment and addresses all specified business needs. System testing is conducted by an independent testing group to provide an unbiased assessment of the product's readiness for production.

1.2.4. Acceptance Testing

Acceptance testing is the final stage before a product is released. This level assesses whether the system meets the end-user and business requirements, and determines its readiness for deployment. During acceptance testing, users or stakeholders validate that the system functions as intended in real-world scenarios. Successful completion of this phase indicates that the software is ready to be deployed into production environments. Typically, acceptance testing includes user acceptance testing (UAT) and sometimes business acceptance testing (BAT), depending on project scope.

1.2.5. Definition of Done (DoD)

To maintain a high level of quality throughout the testing and development process, a strict Definition of Done (DoD) is applied. A feature or user story is only considered complete when all of the following criteria are met:

- Code must be reviewed: Every piece of code must undergo peer review to ensure it meets coding standards, is understandable, maintainable, and free from obvious defects.
- All unit tests must pass: The associated unit tests must successfully pass, confirming that the functionality works as intended at the component level.
- Product Owner (PO) accepted: The Product Owner must review and formally accept the implementation based on agreed acceptance criteria.
- No critical bugs: Critical bugs, which are defects that severely affect core functionality, cause system crashes, or result in data loss, must be fully resolved.
- Minor bugs: Minor bugs—defects that have limited impact on functionality, user experience, or performance—may be documented for future improvement if they are deemed acceptable by the stakeholders, provided they do not hinder critical operations.
- Critical bug definition: A critical bug is any issue that blocks major functionality, causes system failures, compromises data integrity, or poses significant security risks.

1.3 Testing Types

- **Function testing:** is a type of test and mainly involves black box testing. The testing occurs by providing input from the user and checking for expected output. This testing checks the User interface, APIs, Database, and Server/Client communication. The test can be done manually or automatically.
- -**Non-function testing:** is not related to any function or feature of the system, it is used to test project performance or security. There are testing types such as performance testing, load testing, stress testing, usability testing, maintainability testing, and reliability testing.

2. Test Strategy

2.1 Testing Stages

Type of Test	**Stage of Test**			
	Unit	***Integration***	***System***	***Acceptance***
Function Test	X	X	X	X
User Interface test		X	X	X
Data integrity test	X	X	X	X

Table 50 - Testing stages

2.2 Resources

2.2.1. Human Resources

Worker/Doer	**Role**	**Specific Responsibilities/Comments**
Trương Sỹ Quảng	Team leader	System testing, and Acceptance testing
Vũ Nhật Hào	Team member	Unit testing, Integration testing, System testing, and Acceptance testing

Nguyễn Hoàng Bảo	Team member	Unit testing, Acceptance testing, Integration testing, and System testing
Nguyễn Hưng Hảo	Team member	Unit testing, Acceptance testing, and System testing
Tống Trường Thanh	Team member	System testing and Acceptance testing

Table 51 - Testing stages

2.2.2. Environment

Purpose	Tool	Provider	Version
Postman makes API development easy. Our platform offers the tools to simplify each step of the API building process and streamlines collaboration so you can apply this for unit testing	Postman	Postman	v 7.35.0
Chrome is a platform for our web application to run on it.	Chrome	Google	v 86.0.4240.198

Table 51 - Testing stages

2.3 Test Milestones

Milestone Task	Effort(md)	Start Date	End Date
Unit test API	7	25/02/2025	23/03/2025
Functional testing in front-end	7	20/03/2025	10/04/2025
System test	7	8/04/2025	22/04/2025
User acceptance test	7	12/04/2023	22/04/2025

Table 52 - Testing stages

2.4 Deliverables

No	Deliverables	Due Date
1	Fully report of API restful unit test	22/04/2025
2	Document of unit test	22/04/2025
3	Report of supervisor and integration test with front-end web application	22/04/2025
4	Completely report of testing	22/04/2025

Table 53 - Testing stages

3. Test Cases

-The fully unit test cases and details are described in Report5_Unit Test Case.XLSX.
-The module test cases are presented in Report5_Test Case Document.XLSX.

4. Test Reports

4.1 Unit test

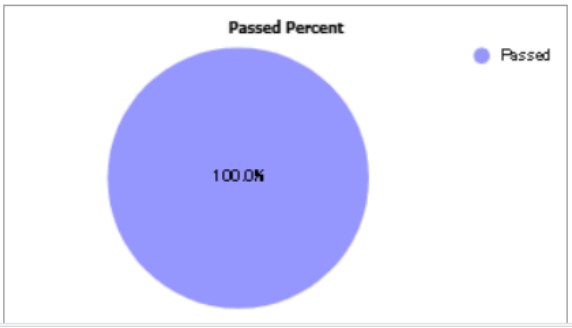
SP25SE030 - OpsLink - Unit Test

File Edit View Insert Format Data Tools Extensions Help

J45

No	Function Code	Passed	Failed	Untested	N	A	B	Total Test Cases
1	Create Reservation	6	0	0	4	2	0	6
2	Assign Table to Reservation	7	0	0	4	3	0	7
3	Check in Reservation	11	0	0	7	4	0	11
4	Create Order	6	0	0	4	2	0	6
5	Add Food to Order	10	0	0	6	4	0	10
6	Check Out Order	15	0	0	9	3	3	15
7	Cancel Check Out Order	13	0	0	10	2	1	13
8	Cooking OrderItem	12	0	0	10	2	0	12
9	Done Cooking OrderItem	12	0	0	10	2	0	12
10	Done Serving OrderItem	12	0	0	10	2	0	12
11	Create Payment QR	6	0	0	3	3	0	6
12	Create Payment Cash	6	0	0	3	3	0	6
13	Create Workshop	13	0	0	9	3	1	13
14	Register Workshop	12	0	0	9	3	0	12
15	Check In Workshop	5	0	0	1	4	0	5
16	Assign Table for Workshop Register	6	0	0	1	5	0	6
17	Register Working Slot	14	0	0	5	5	4	14
18	Approve Working Slot Registration	3	0	0	1	1	1	3
19	Assign Staff to Zone Schedule	7	0	0	1	5	1	7
20	Auto Assign Staff to Zone Schedule	10	0	0	4	3	3	10
	Sub total	186	0	0	111	61	14	186

Test coverage	**100.00 %**
Test successful coverage	**100.00 %**
Normal case	**59.68 %**
Abnormal case	**32.80 %**
Boundary case	**7.53 %**



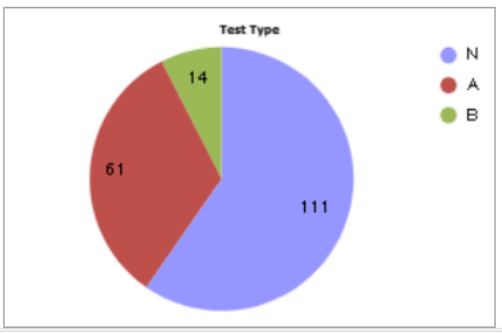


Figure 137- Unit test

4.2 Test case

SP25SE030 - OpsLink - TEST REPORT DOCUMENT

Saved to Drive

File Edit View Insert Format Data Tools Extensions Help

TEST STATISTICS

Project Name	*Service Link System to enhance operations in Pizza restaurant combined with workshop model*	**Creator**		Nguyễn Hưng Hảo, Vũ Nhật Hảo, Hoàng Bảo
Project Code	*SP25SE030*	**Reviewer/Approver**		Nguyễn Nguyên Bình
Document Code	*SP25SE030_Test Report_v1.0*	**Issue Date**		*05 Apr 2025*
Notes				

No	Module code	Passed	Failed	Pending	N/A	Number of test cases
1	Create Reservation	8	0	0	0	8
2	Assign Table To Reservation	7	0	0	0	7
3	Check In Table	4	0	0	0	4
4	Create Order	6	0	0	0	6
5	Add Food To Order	11	0	0	0	11
6	Check Out Order	10	0	0	0	10
7	Cancel Check Out	11	0	0	0	11
8	Cooking OrderItem	5	0	0	0	5
9	Serving OrderItem	5	0	0	0	5
10	Done OrderItem	5	0	0	0	5
11	Create Payment QR	13	0	0	0	13
12	Create Payment Cash	14	0	0	0	14
13	Create Workshop	13	0	0	0	13
14	Cancel Workshop	7	0	0	0	7
15	Register Workshop	11	0	0	0	11
16	View Workshops	7	0	0	0	7
17	Check In Workshop	6	0	0	0	6
18	Assign Table Register	12	0	0	0	12
19	Create Staff Account	8	0	0	0	8
20	Create Working Slot	6	0	0	0	6
21	Register Working Slot	11	0	0	0	11
22	Approve Working Slot Registration	4	0	0	0	4
23	Reject Working Slot Registration	4	0	0	0	4
24	Assign Staff to Zone Schedule	8	0	0	0	8
25	Auto Assign Staff to Zone Schedule	8	0	0	0	8
26	Swap Working Slots	8	0	0	0	8
27	Approve Swap	6	0	0	0	6
28	Pending Swap	3	0	0	0	3
29	Reject Swap	4	0	0	0	4
	Sub total	225	0	0	0	225

Test coverage 100.00 %

Test successful coverage 100.00 %

Figure 138- Test case

VI. Release Package & User Guides

1. Deliverable Package

No.	Deliverable Item	Description
1	Project Schedule/Tracking	Modify
2	Project Backlog	Modify
3	Source Codes	Modify
4	Database Script(s)	Modify
5	Final Report Document	New
6	Test Cases Document	New
7	Defects List	New
8	Issues List	New
9	Slide	Modify

Table 54 - Testing stages

2. Installation Guides

2.1 System Requirements

Components	Minimum	Recommended
OS	Ubuntu 20.04.05/Windows Server 2019 or above	Ubuntu 20.04.05/Windows Server 2019 or above
Processor	1.4 GHz 64-bit processor	2.4 GHz 64-bit processor or above

Memory	4 GB RAM	8 GB RAM
Network	Broadband Internet connection	Broadband Internet connection
Storage	30GB SSD	50GB SSD or above
Browser	Google Chrome: Version 120	Google Chrome: Version 120 or higher
Mobile Device	Android 4.1	Android 4.1 or higher

2.1.1 Setting up environment at server side

2.1.1.1 Hardware requirement

Hardware	Specification
Internet Connection	LAN Cable, Wi-fi (8Mbps)
Operation System	MacOS/Windows 10
Computer Processor	Intel Core i5 2.5 GHz
Computer Memory	8GB and more

Table 55 - Testing stages

2.1.1.2 Software requirement

Name	Name/Version	Description
Environment	.NET Framework (.NET 6 / .NET 7)	Specification for developing server
Operating System	MacOS/Window 10	Operating system and platform for development

Modelling tool	Draw.io,	Used to design diagram
IDE	Visual Studio Code, Visual Studio 2022	Programming tools
DBMS	SQL Server	Used to create & manage the database for system
Web server	Linux	Deployment environment
Web browser	Chrome 3.6.6 or above	Browser

Table 55 - Testing stages

2.2 Setup Files

Application	**URL**
Server Back-end	https://github.com/haovn2411/pizza-service
ReactJS Front-end	https://github.com/MeoKool/Capstone-project-order-pizza-chef https://github.com/MeoKool/Capstone-project-order-pizza-staff https://github.com/Tom-Dev1/Capstone-project-order-pizza-manager https://github.com/Tom-Dev1/Capstone-project-order-pizza-user https://github.com/Tom-Dev1/capstone-prj-pizza-customer

Table 55 - Testing stages

2.3 Installation Instruction

2.3.1 Environment installation

- Visual Studio (code editor): download and install via link https://code.visualstudio.com/download

2.3.2 Back-end Installation Guide (C# with .NET Framework)

The back-end of the application is built with Java. Follow these steps to set up the back-end environment:

Step 1: Install Visual Studio 2022

- Download Visual Studio 2022 from the official Microsoft website.
- During installation, select the ".NET desktop development" workload to ensure all necessary components for C# development are included.

Step 2: Install .NET Framework (if not already installed)

- Visual Studio 2022 usually installs the needed .NET Framework automatically.
- If required, you can manually download and install the correct version (e.g., .NET Framework 4.7.2 or newer) from the Microsoft .NET Framework downloads.

Step 3:Set up SQL Server

- Install SQL Server (e.g., SQL Server 2019 Express) from Microsoft SQL Server downloads.
- Also, install SQL Server Management Studio (SSMS) for database management.

Step 4:Clone the back-end repository from GitHub

- Open a terminal (e.g., Command Prompt, PowerShell, or Git Bash) and run the following command:[*git clone git clone https://github.com/Van02022001/SCPS-IMS.git*]

Step 5: Clone the back-end repository from GitHub

- Use the command `git clone [https://github.com/Van02022001/SCPS-IMS.git]` to clone the back-end repository to your local machine.

Step 6: Open the project in Visual Studio 2022

- Launch Visual Studio 2022.
- Click "Open a project or solution", and navigate to the cloned project folder.
- Select the .sln (Solution) file to open the project.

Step 7: Restore Dependencies

- Open the terminal in your IDE and run `mvn clean install` to resolve and download all dependencies and build the project.
-

Step 8: Configure the application

- Once the project is loaded, right-click the solution in Solution Explorer and select "Restore NuGet Packages".
- This will download and set up all required dependencies for the project.

Step 9: Configure the application

- Locate the appsettings.json or Web.config file (depending on the project type).
- Update database connection settings such as:
- **Server (your SQL Server name)**
- **Database (your database name)**
- **User Id and Password**
- Example connection string:
- *"ConnectionStrings": { "DefaultConnection": "Server=localhost;Database=YourDatabase;User Id=yourusername;Password=yourpassword;"}*

Step 10:Build and run the application

- Verify the back-end service
- Once running, check the back-end services via a web browser.
- Typically, the API or welcome page will be available at: *http://localhost:5000/swagger*

2.3.3 Front-end installation

Step 1: Download code package in GitHub via link

Step 2: open project with terminal (macOS) or cmd (Win10)

- Redirect to the path of the project (using cd *path)
- Run an app in Visual Studio (*recommended*) (using code .)

Step 3: Open terminal in Visual Studio and Install all modules with command line *"**npm install**"*.

Step 4: Run *"**npm run dev**"* to start the project.

3. User Manual

3.1 Overview

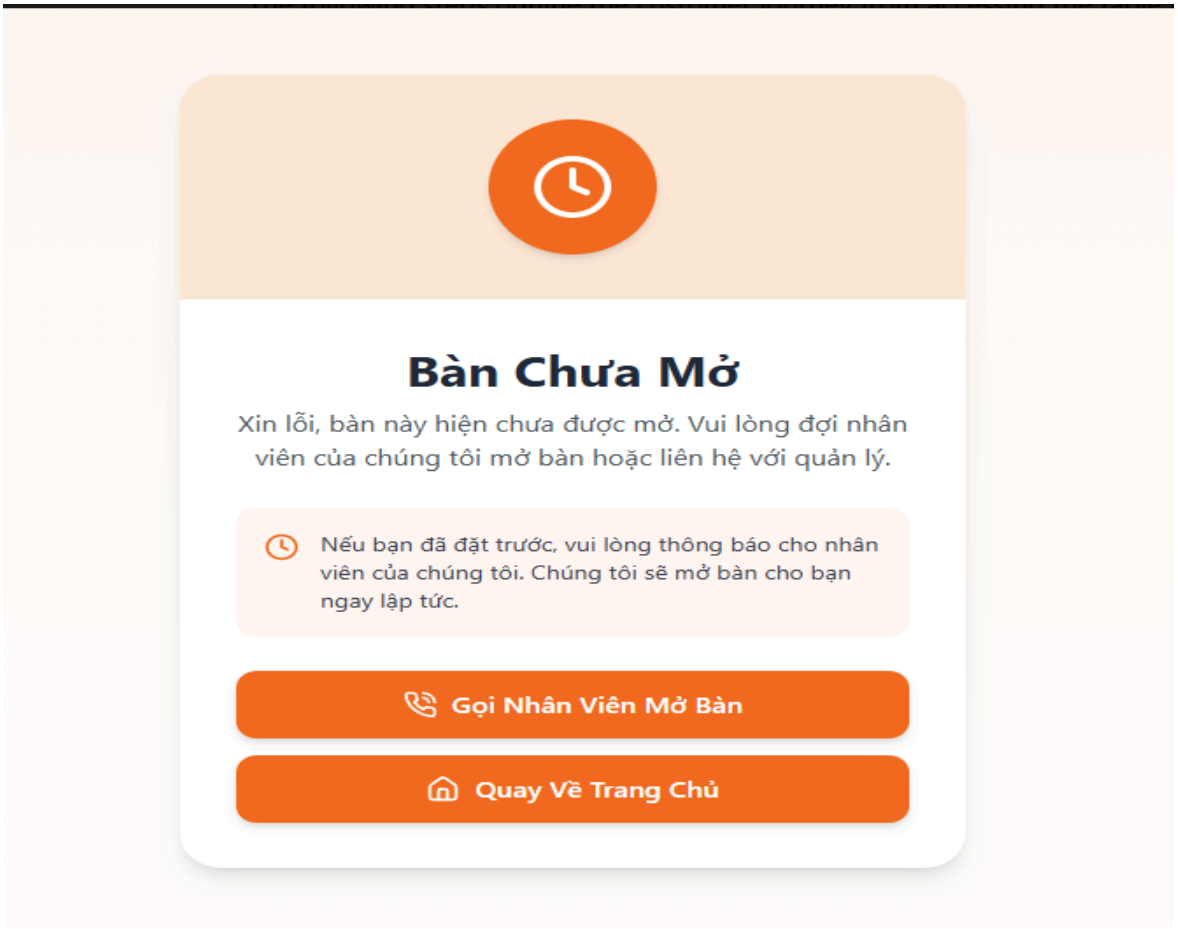
This user manual describes the key business workflows supported by the restaurant platform. Each workflow lists the actors, prerequisites, primary steps, alternative flows, and post-conditions so that staff, managers, and customers can clearly understand how to use the system.

3.2 Workflow 1 – In-Store Ordering via Table QR

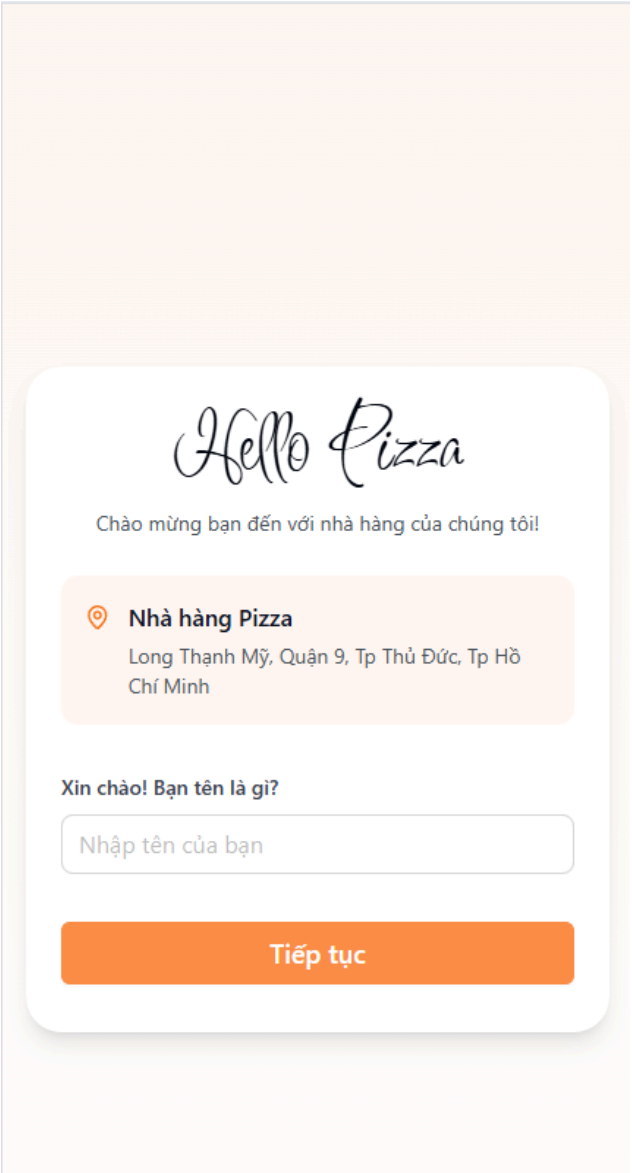
Item	Description
Purpose	Allow dine-in guests to open a table, place orders, and complete payment via QR-code scanning.
Actors	Customer, Waiter/Waitress (Staff), Kitchen Staff, Cashier / Manager.
Prerequisites	• The table status is Closed. • A unique QR-code linked to the table ID is available on the table. • Staff has the "Open Table" permission.
Main Flow	

1. Customer scans the QR on a Closed table

2. App displays **“Gọi nhân viên mở bàn”**

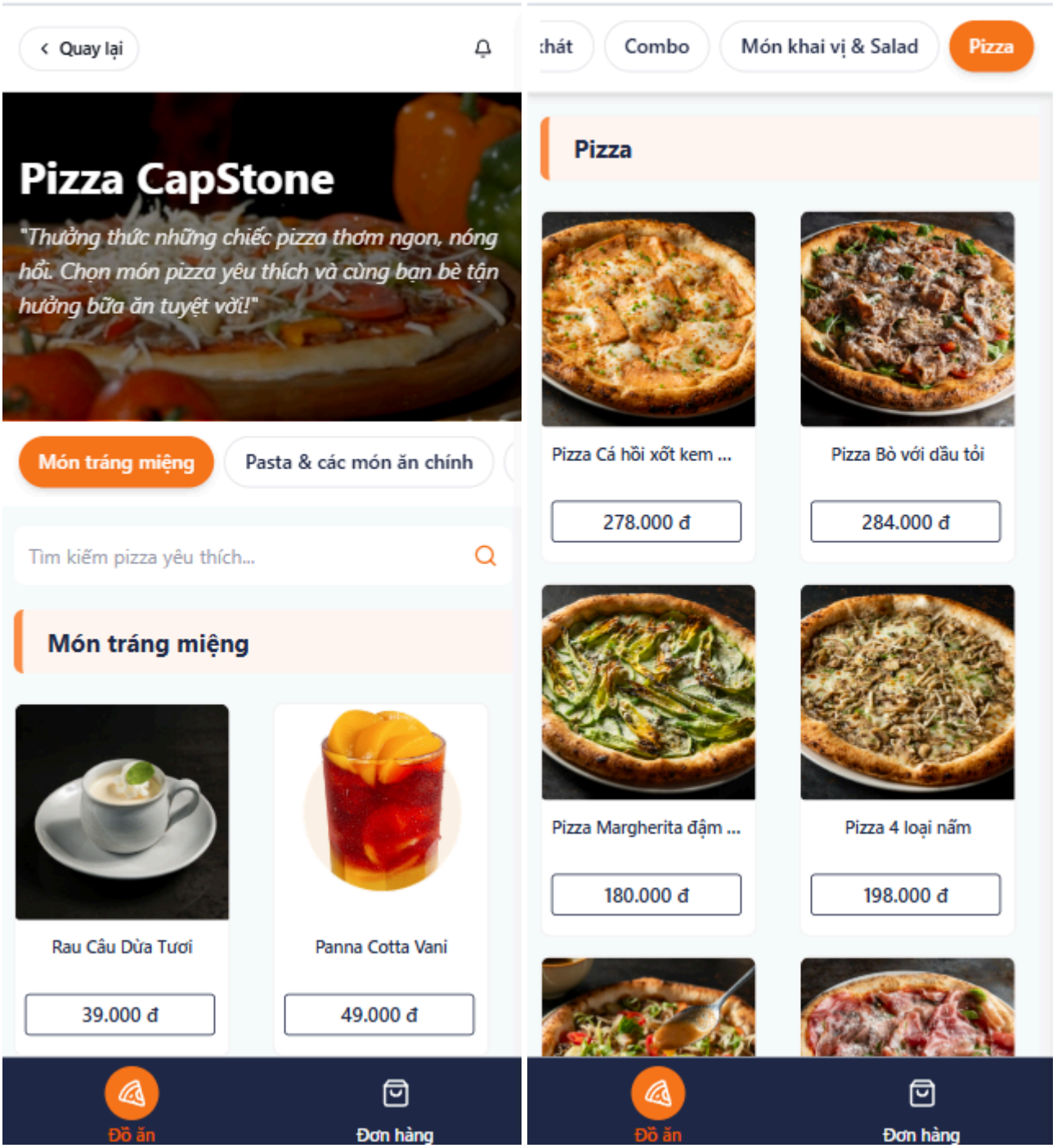


3. Staff taps **“Mở bàn”** → Table status becomes Open

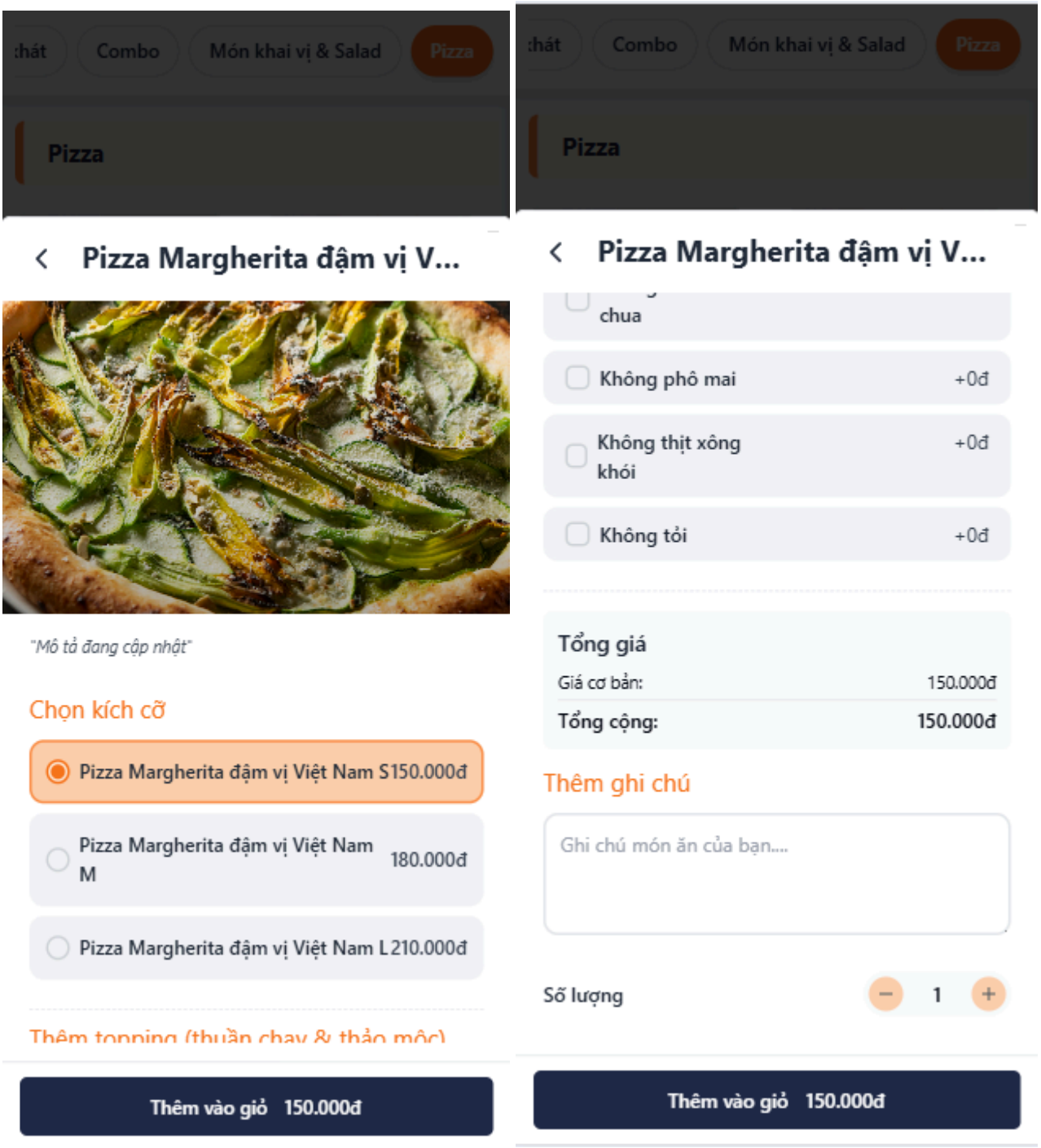


4. Customer rescans the QR → Digital Menu appears

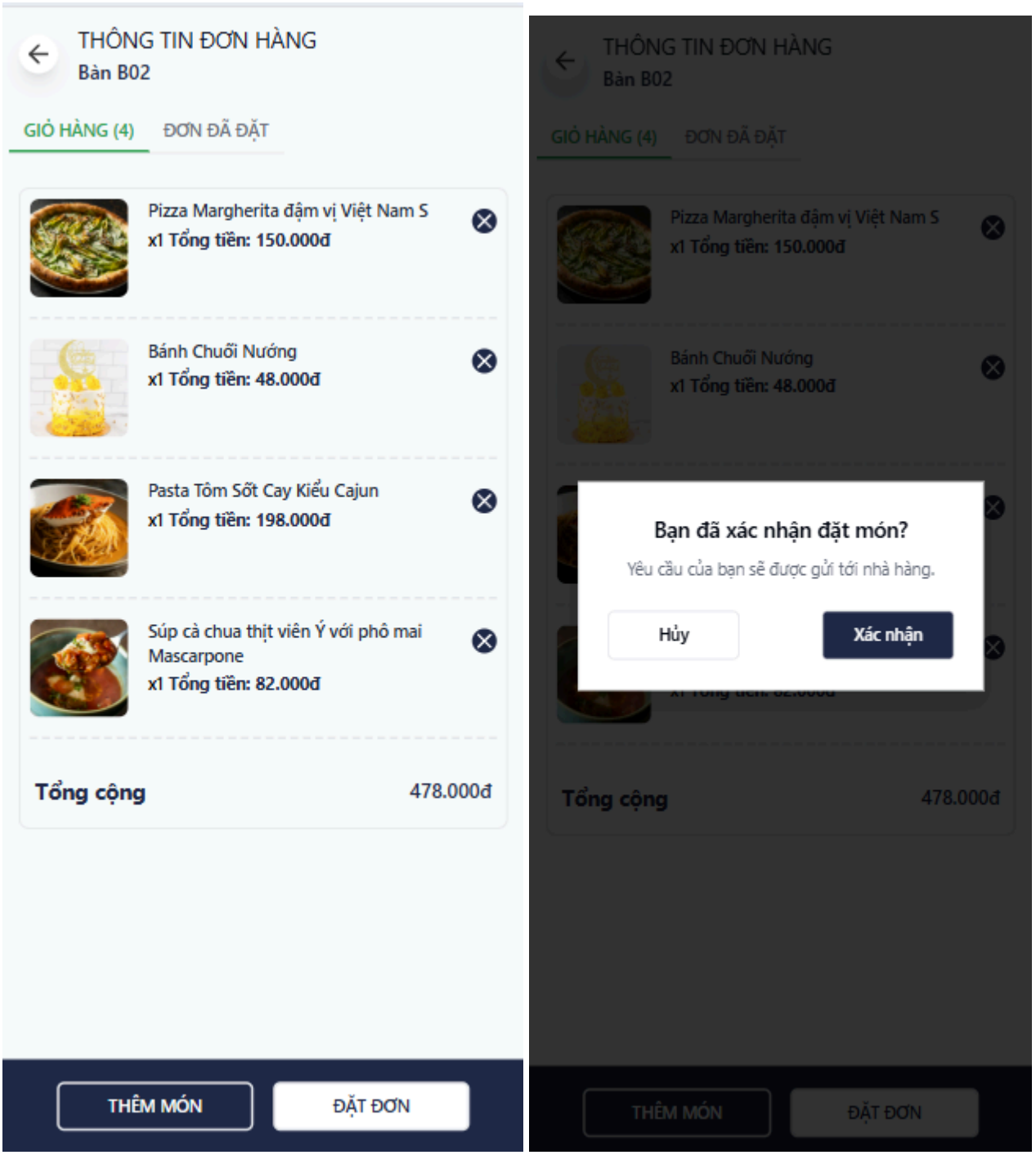

Nhà hàng Pizza
Long Thạnh Mỹ, Quận 9, Tp Thủ Đức, Tp Hồ Chí Minh
Pizza
HOT & REFRESH
ORDER NOW
50% OFF
Double The Fun On Every Delivery
Chào buổi tối Oplinks
Chúng tôi sẽ trả đồ cho bạn tại bàn: B02
Số điện thoại của bạn sẽ được tích điểm !
Gọi thanh toán
Gọi nhân viên
Cần có đơn hàng để đánh giá
Đánh giá
Xem Menu - Gọi món

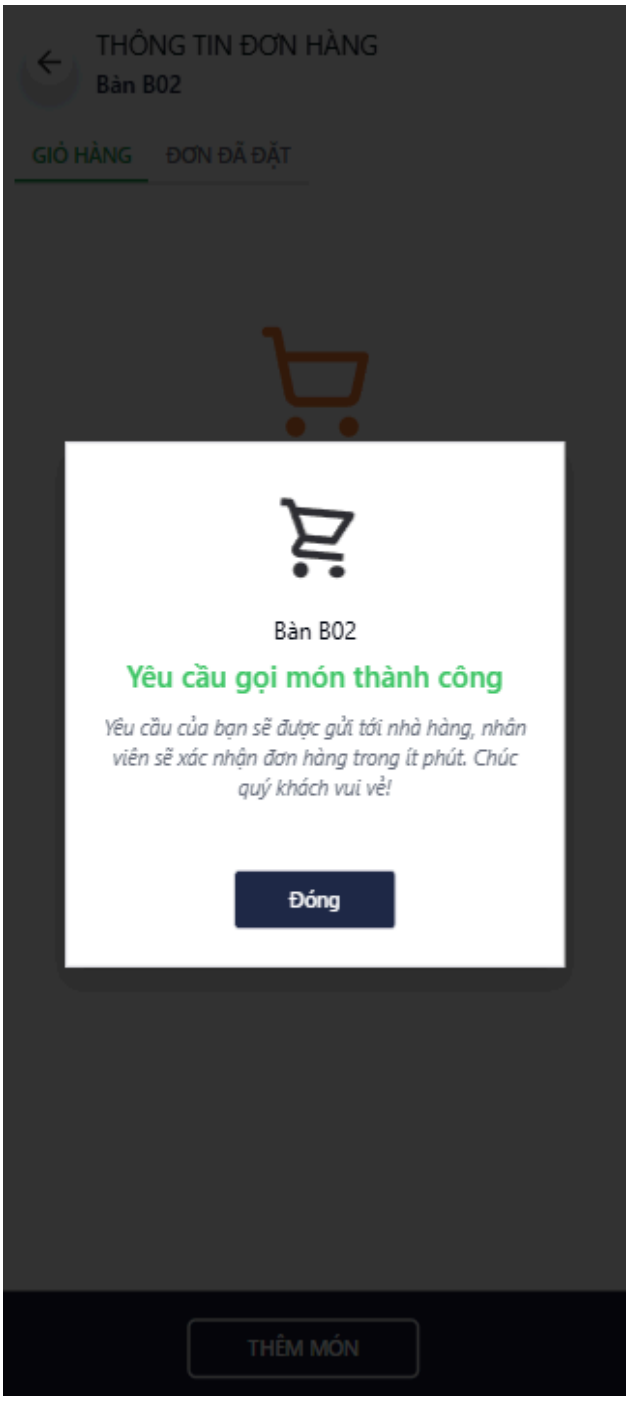


5. Customer taps **"Thêm vào giỏ"** in Item Details

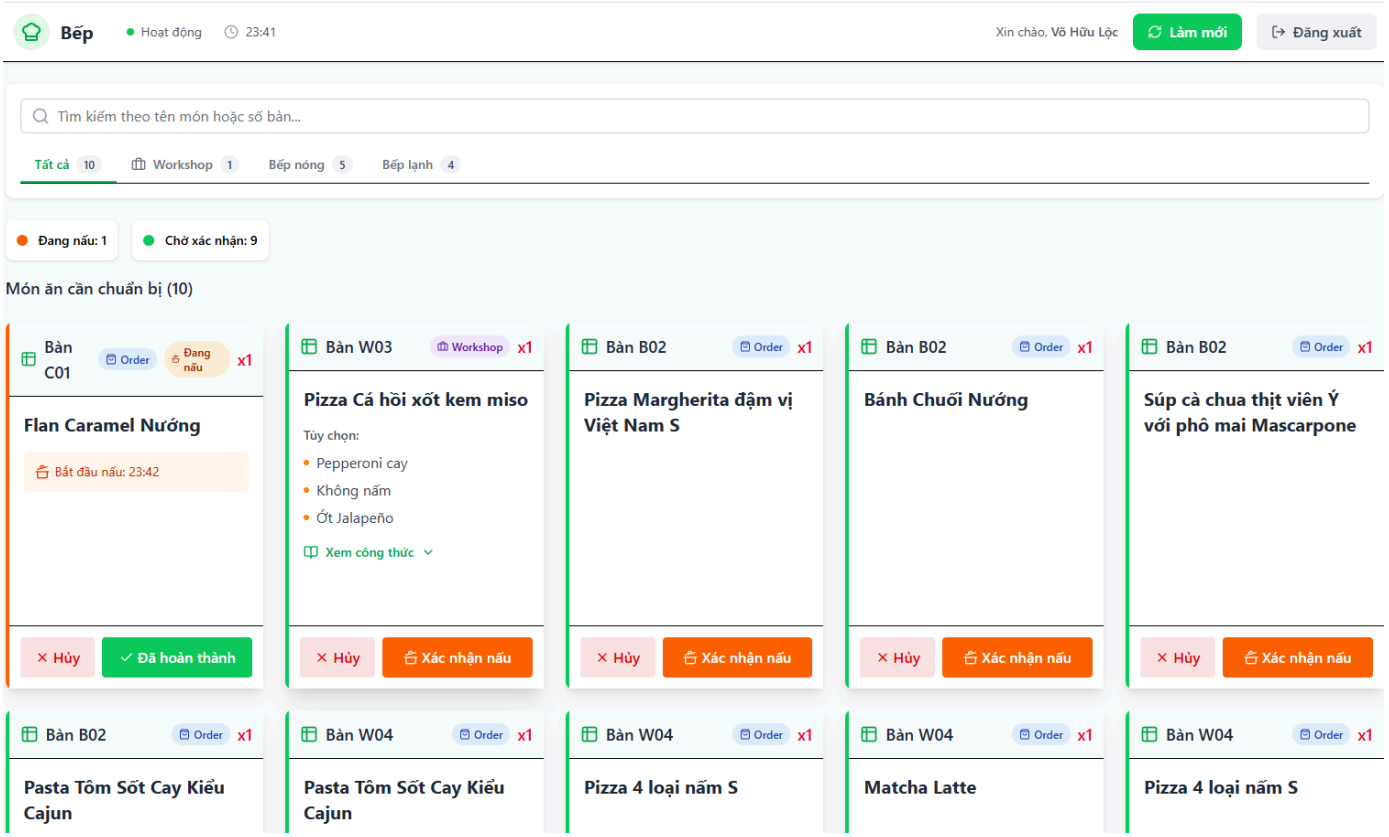


6. Customer taps **“Đặt đơn”** → Order items sent to Kitchen

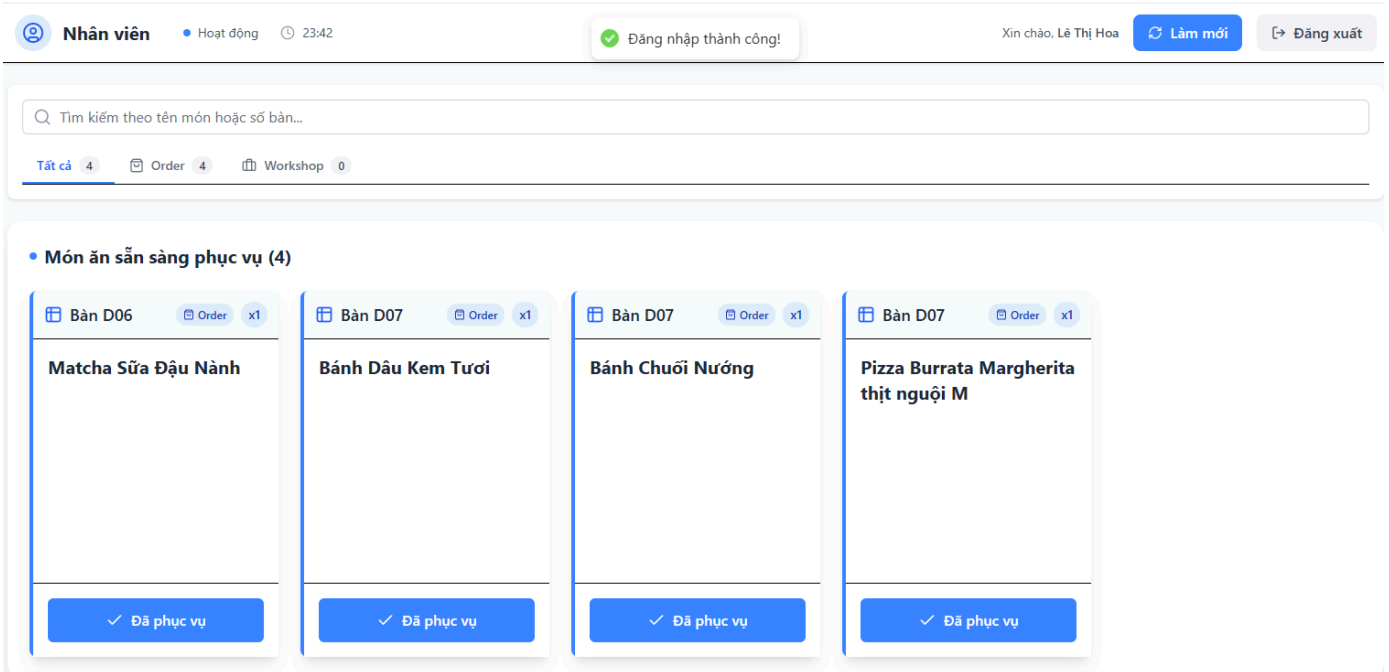

THÔNG TIN ĐƠN HÀNG
Bàn B02
GIỎ HÀNG (4)
ĐƠN ĐÃ ĐẶT
Pizza Margherita đậm vị Việt Nam S
x1 Tổng tiền: 150.000đ
Bánh Chuối Nướng
x1 Tổng tiền: 48.000đ
Pasta Tôm Sốt Cay Kiểu Cajun
x1 Tổng tiền: 198.000đ
Súp cà chua thịt viên Ý với phô mai Mascarpone
x1 Tổng tiền: 82.000đ
Tổng cộng
478.000đ
THÊM MÓN
ĐẶT ĐƠN
THÔNG TIN ĐƠN HÀNG
Bàn B02
GIỎ HÀNG (4)
ĐƠN ĐÃ ĐẶT
Pizza Margherita đậm vị Việt Nam S
x1 Tổng tiền: 150.000đ
Bánh Chuối Nướng
x1 Tổng tiền: 48.000đ
Bạn đã xác nhận đặt món?
Yêu cầu của bạn sẽ được gửi tới nhà hàng.
Hủy
Xác nhận
Tổng cộng
478.000đ
THÊM MÓN
ĐẶT ĐƠN



7. Kitchen Staff marks order Cooking → Complete → Serving

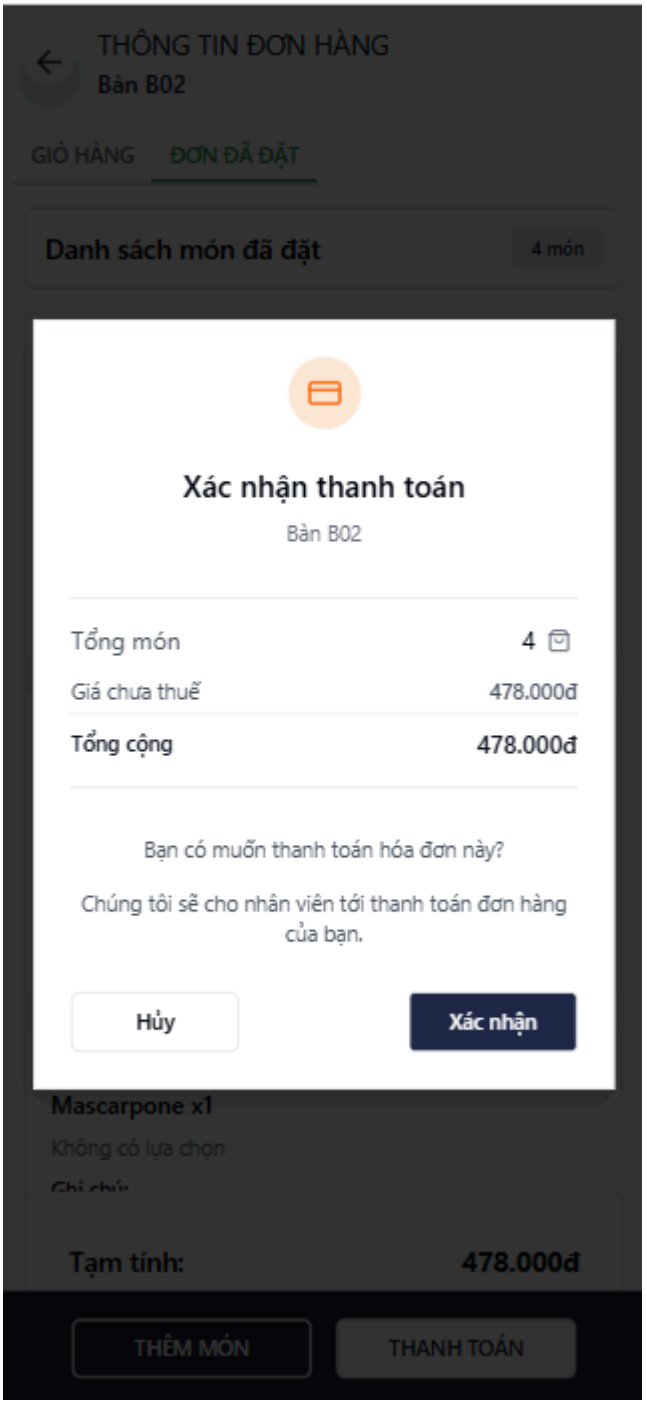


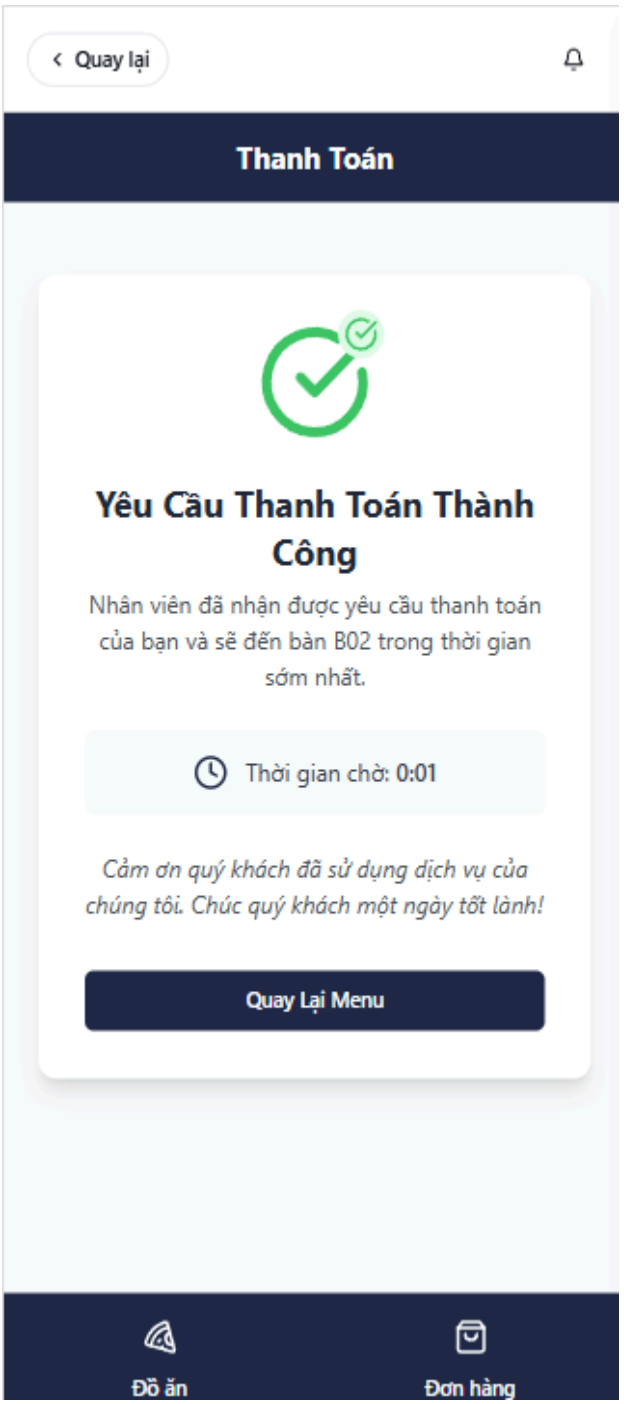
8. Waiter delivers dishes & taps **“Đã phục vụ”** → Order status *Done*



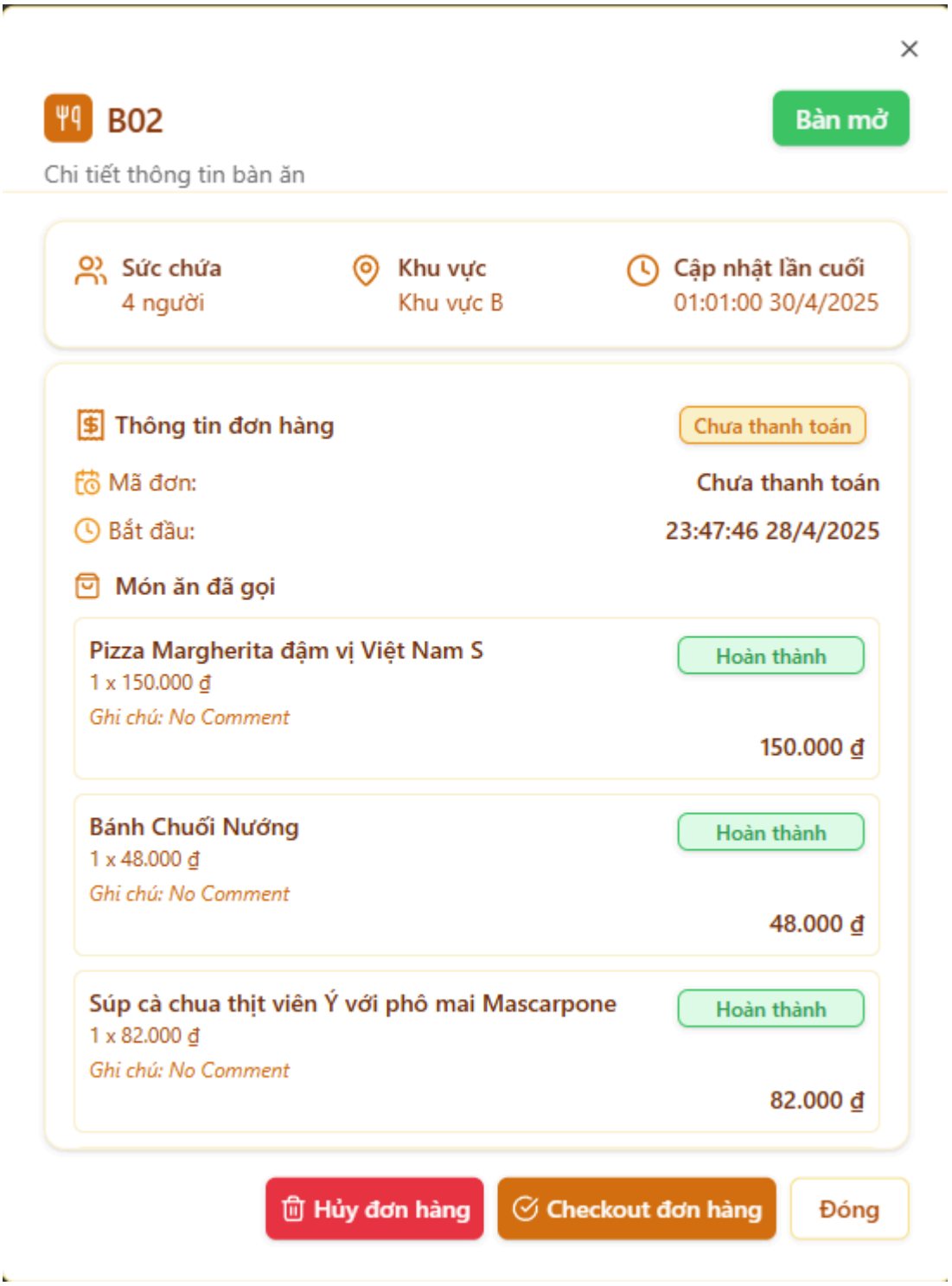
9. Customer taps **“Thanh toán”** after the meal

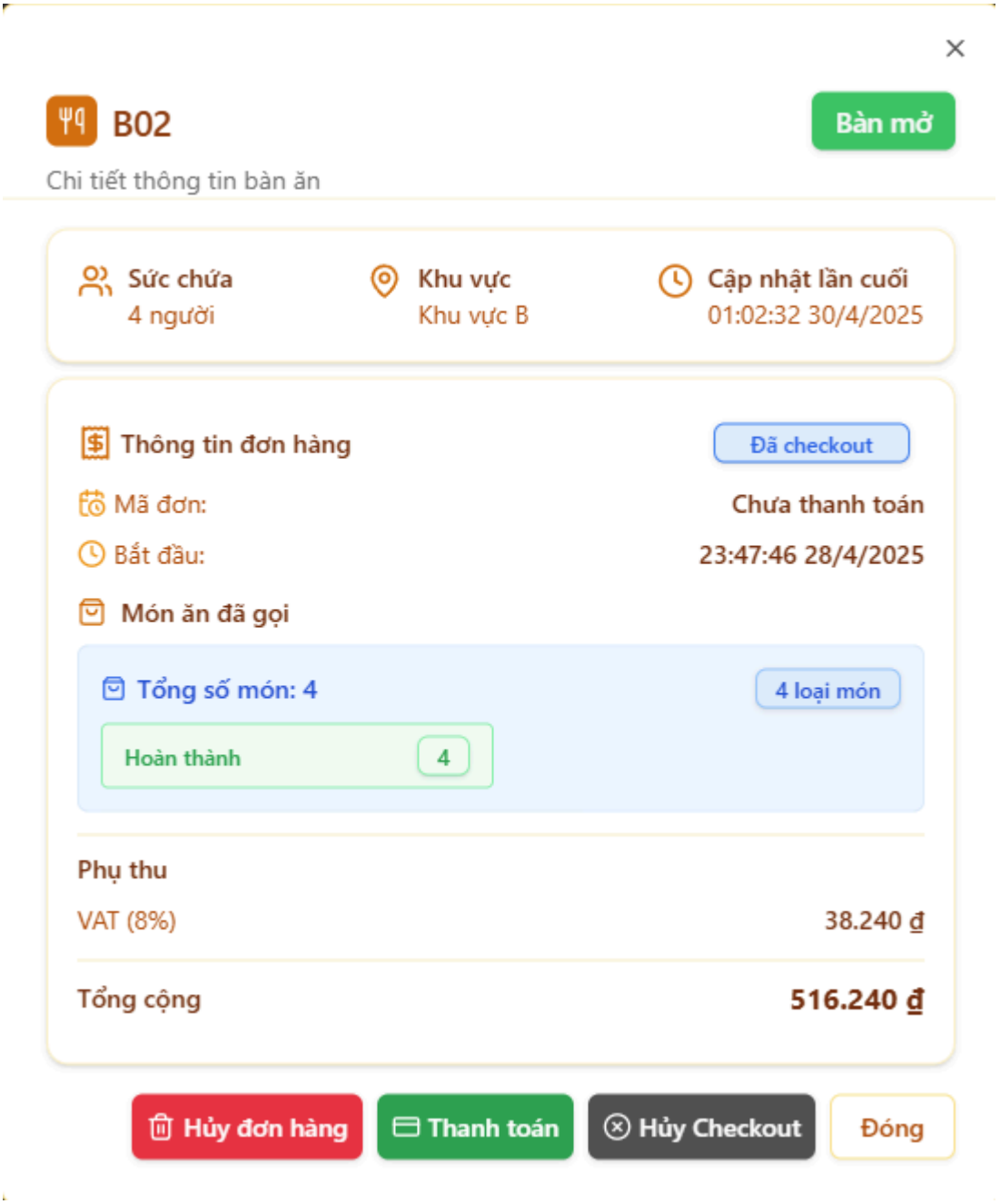

THÔNG TIN ĐƠN HÀNG
Bàn B02
Bánh Chuối Nướng x1
48.000đ
Không có lựa chọn
Ghi chú:
• No Comment
Tổng tiền:
48.000đ
Hoàn thành
Súp cà chua thịt viên Ý với phô mai Mascarpone x1
82.000đ
Không có lựa chọn
Ghi chú:
• No Comment
Tổng tiền:
82.000đ
Hoàn thành
Pasta Tôm Sốt Cay Kiểu Cajun x1
198.000đ
Không có lựa chọn
Ghi chú:
• No Comment
Tổng tiền:
198.000đ
Hoàn thành
Tạm tính:
478.000đ
THÊM MÓN
THANH TOÁN

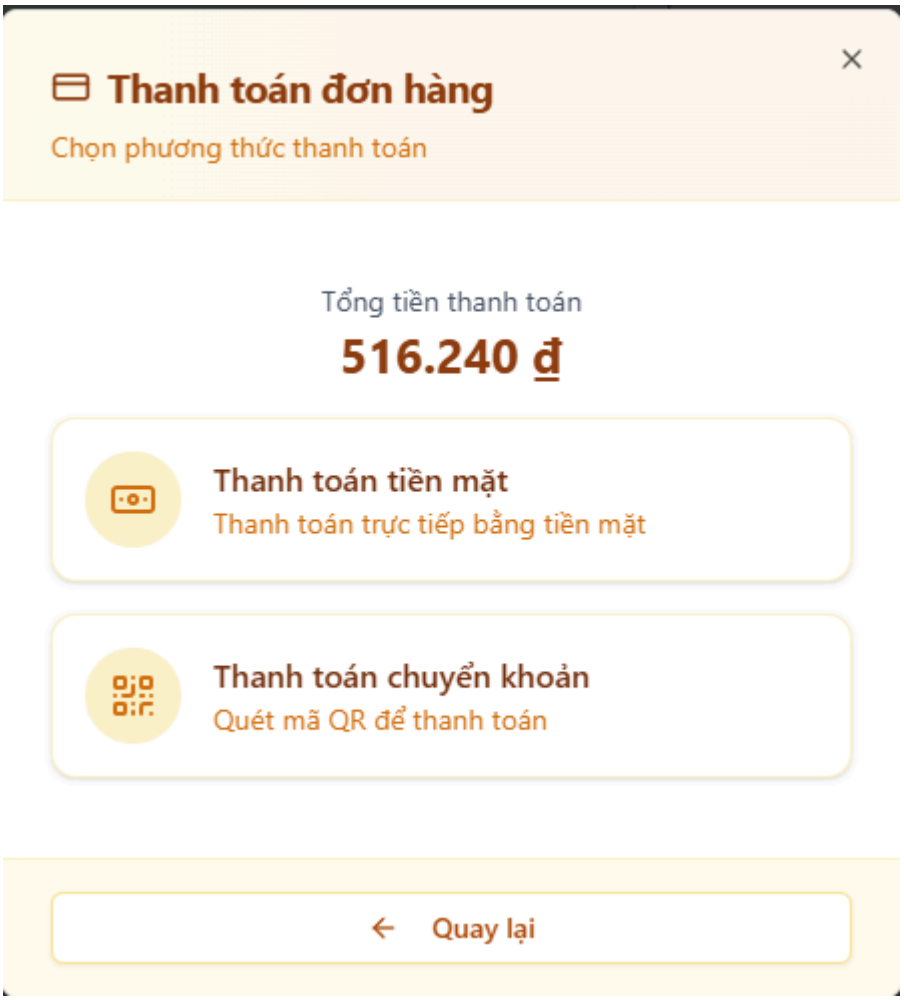




10.Staff or Manager checkout order and selects **“QR Code”** or **“Thanh toán tiền mặt”** & completes payment


B02
Bàn mở
Chi tiết thông tin bàn ăn
Sức chứa
4 người
Khu vực
Khu vực B
Cập nhật lần cuối
01:01:00 30/4/2025
Thông tin đơn hàng
Chưa thanh toán
Mã đơn:
Chưa thanh toán
Bắt đầu:
23:47:46 28/4/2025
Món ăn đã gọi
Pizza Margherita đậm vị Việt Nam S
1 x 150.000 ₫
Ghi chú: No Comment
Hoàn thành
150.000 ₫
Bánh Chuối Nướng
1 x 48.000 ₫
Ghi chú: No Comment
Hoàn thành
48.000 ₫
Súp cà chua thịt viên Ý với phô mai Mascarpone
1 x 82.000 ₫
Ghi chú: No Comment
Hoàn thành
82.000 ₫
Hủy đơn hàng
Checkout đơn hàng
Đóng


B02
Bàn mở
Chi tiết thông tin bàn ăn
Sức chứa
4 người
Khu vực
Khu vực B
Cập nhật lần cuối
01:02:32 30/4/2025
Thông tin đơn hàng
Đã checkout
Mã đơn:
Chưa thanh toán
Bắt đầu:
23:47:46 28/4/2025
Món ăn đã gọi
Tổng số món: 4
4 loại món
Hoàn thành
4
Phụ thu
VAT (8%)
38.240 ₫
Tổng cộng
516.240 ₫
Hủy đơn hàng
Thanh toán
Hủy Checkout
Đóng



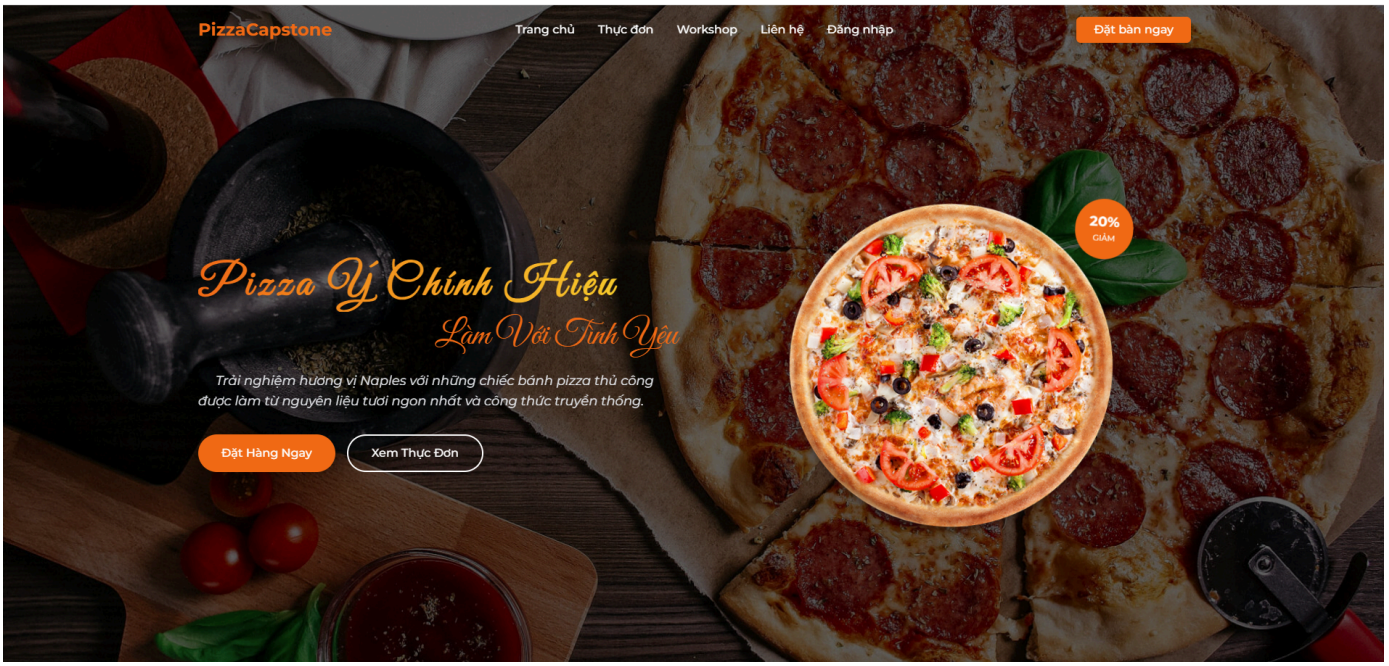
11. Payment success → Order Paid; Table status Closed

[Insert image: W1_11.png]

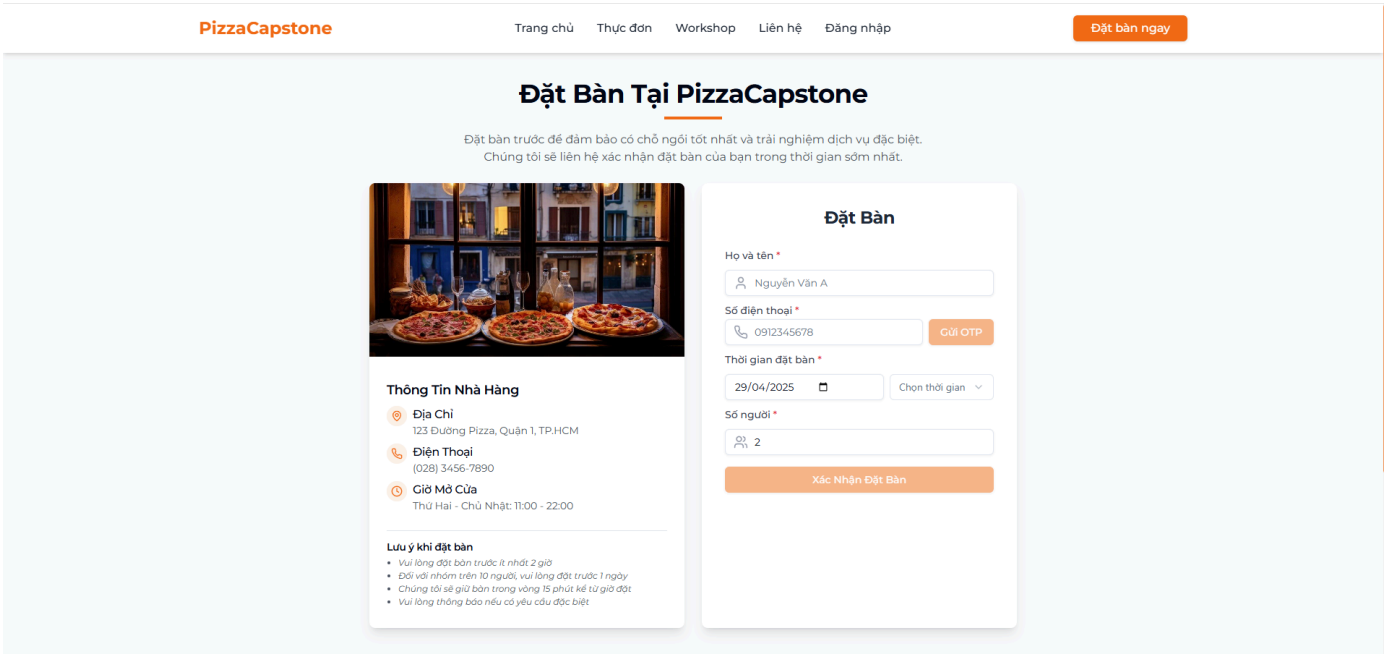
3.3 Workflow 2 – Online Table Reservation

Item	Description
Purpose	Let customers reserve a table in advance via the website.
Actors	Customer, Manager / Host, System Scheduler.
Prerequisites	Web booking form is accessible. SMS OTP service is active.

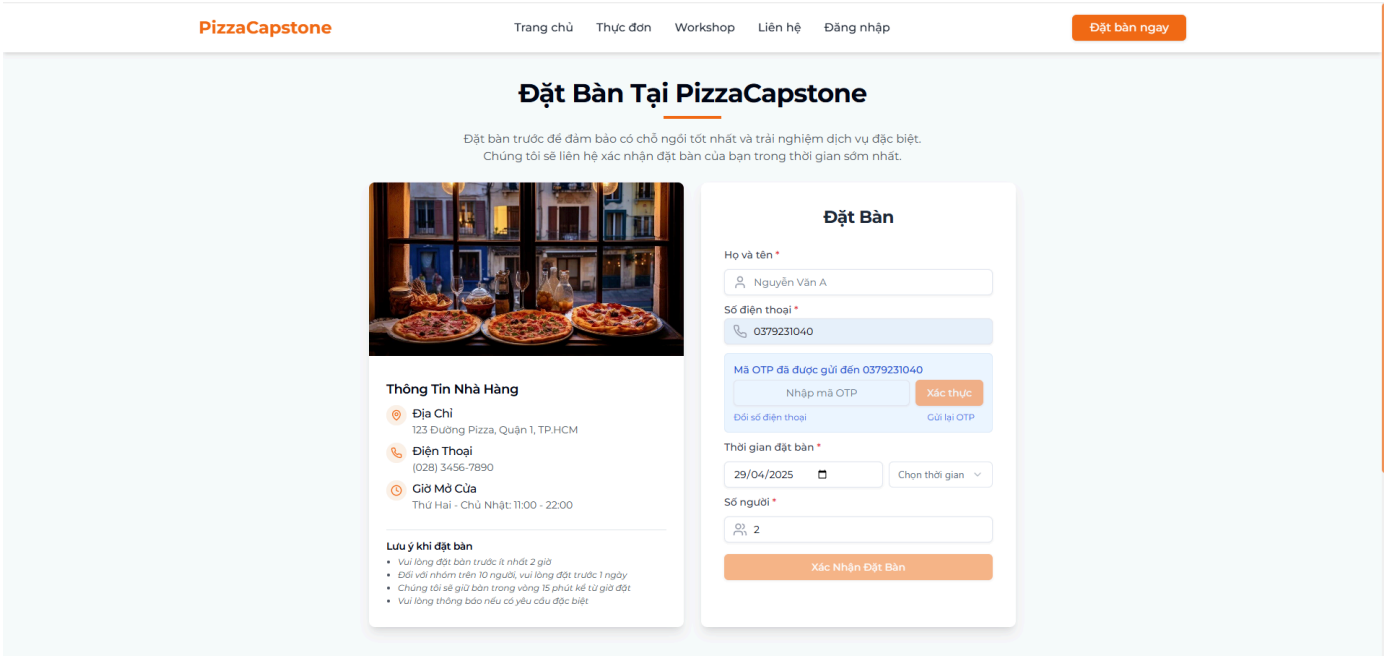
12. Customer opens the website & clicks **"Đặt bàn ngay"**

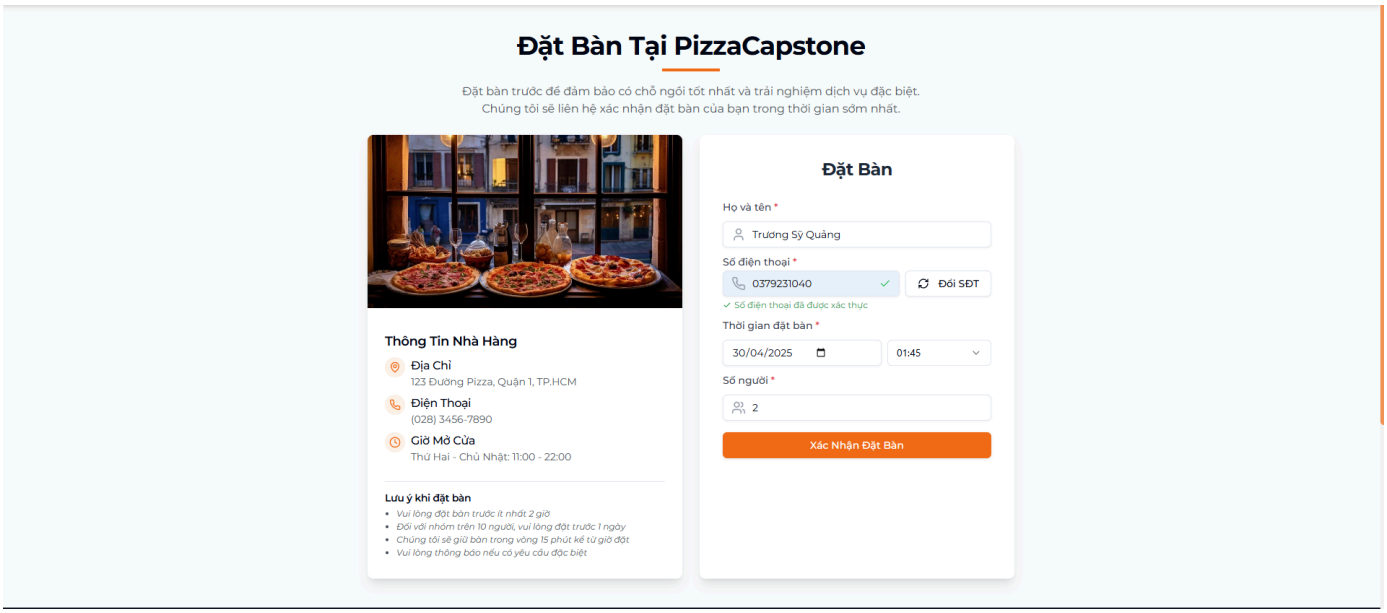


13. Fills Name, Phone, Date & Time, Party Size

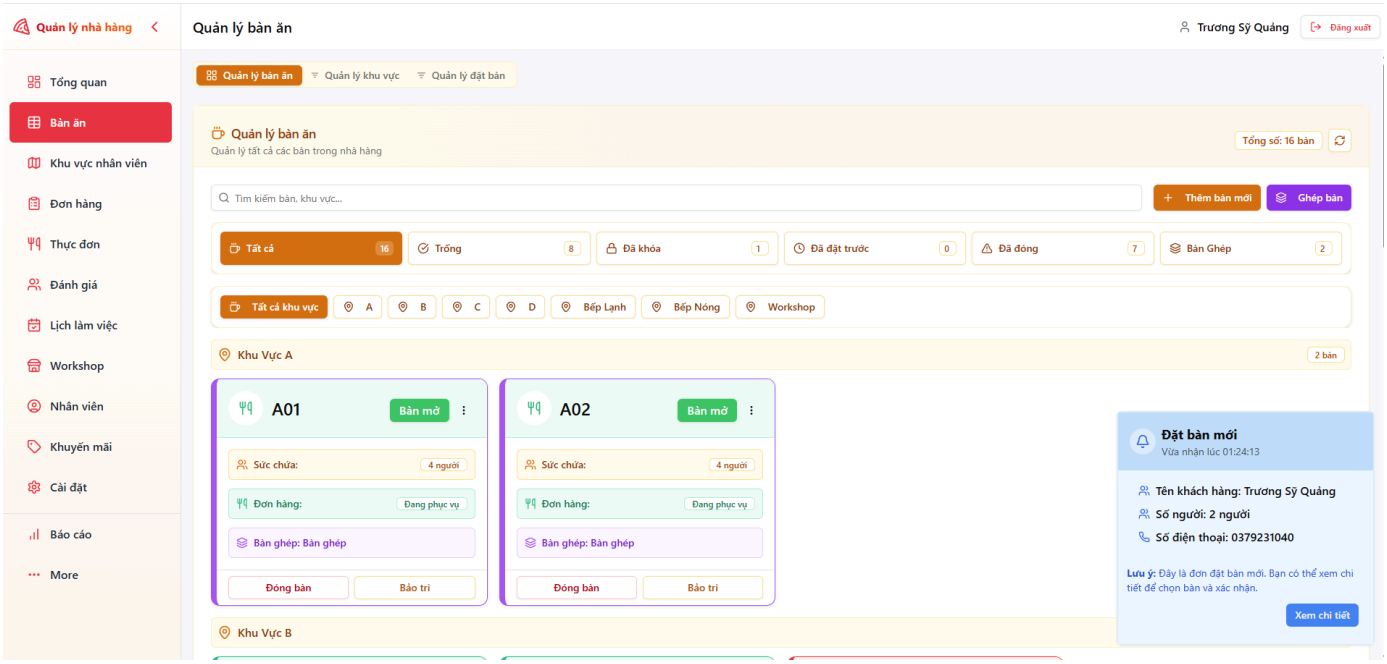


14. Clicks **“Gửi mã OTP”**, enters code, then clicks **“Xác nhận đặt bàn”**

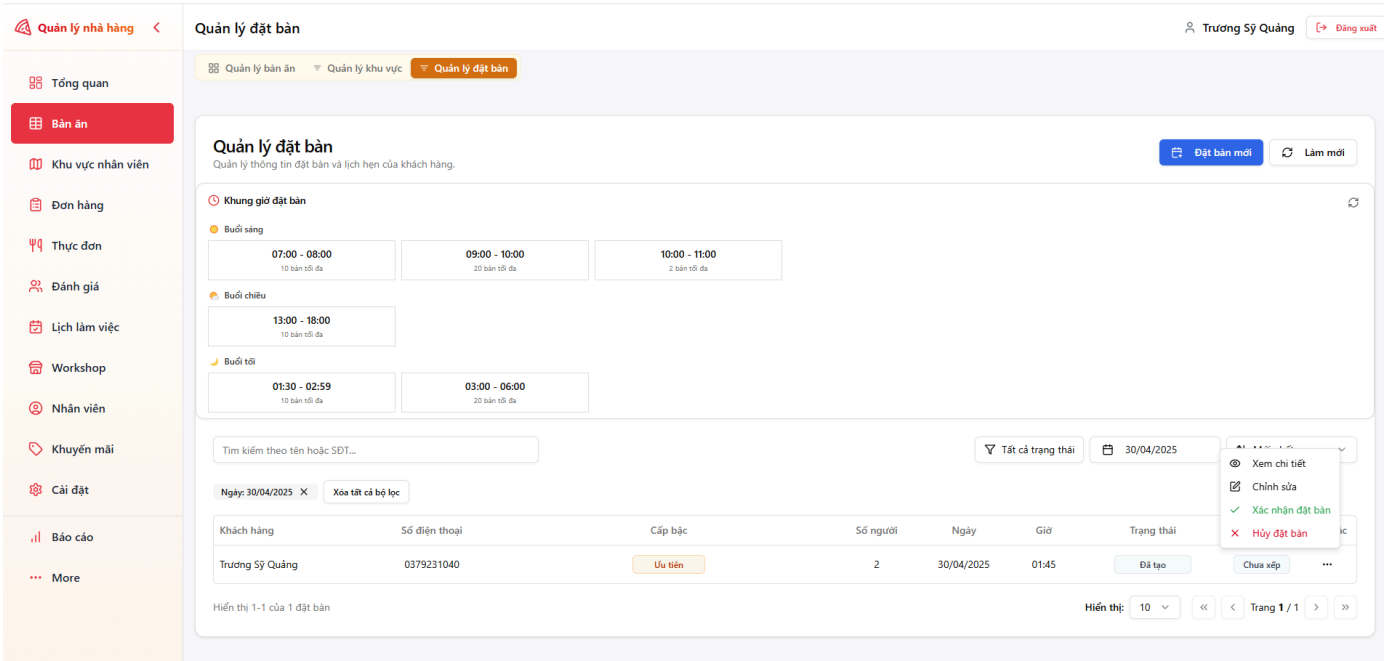




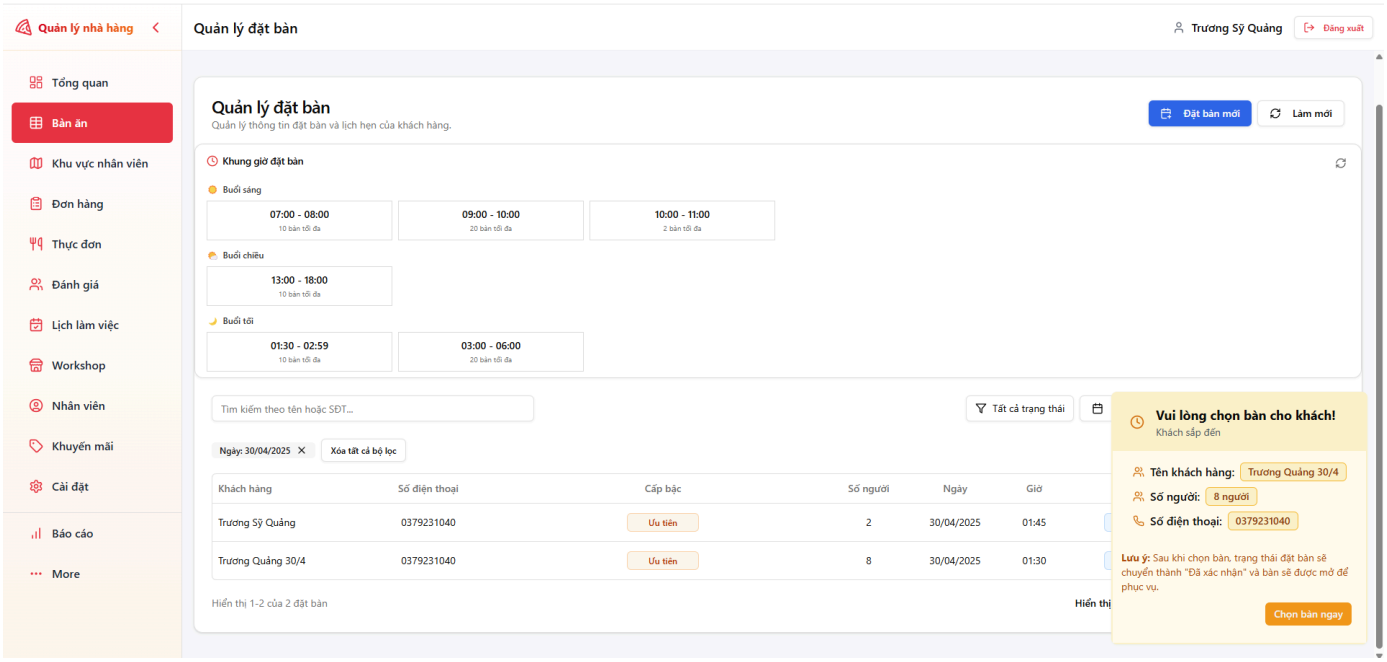
15. Manager receives New reservation notification



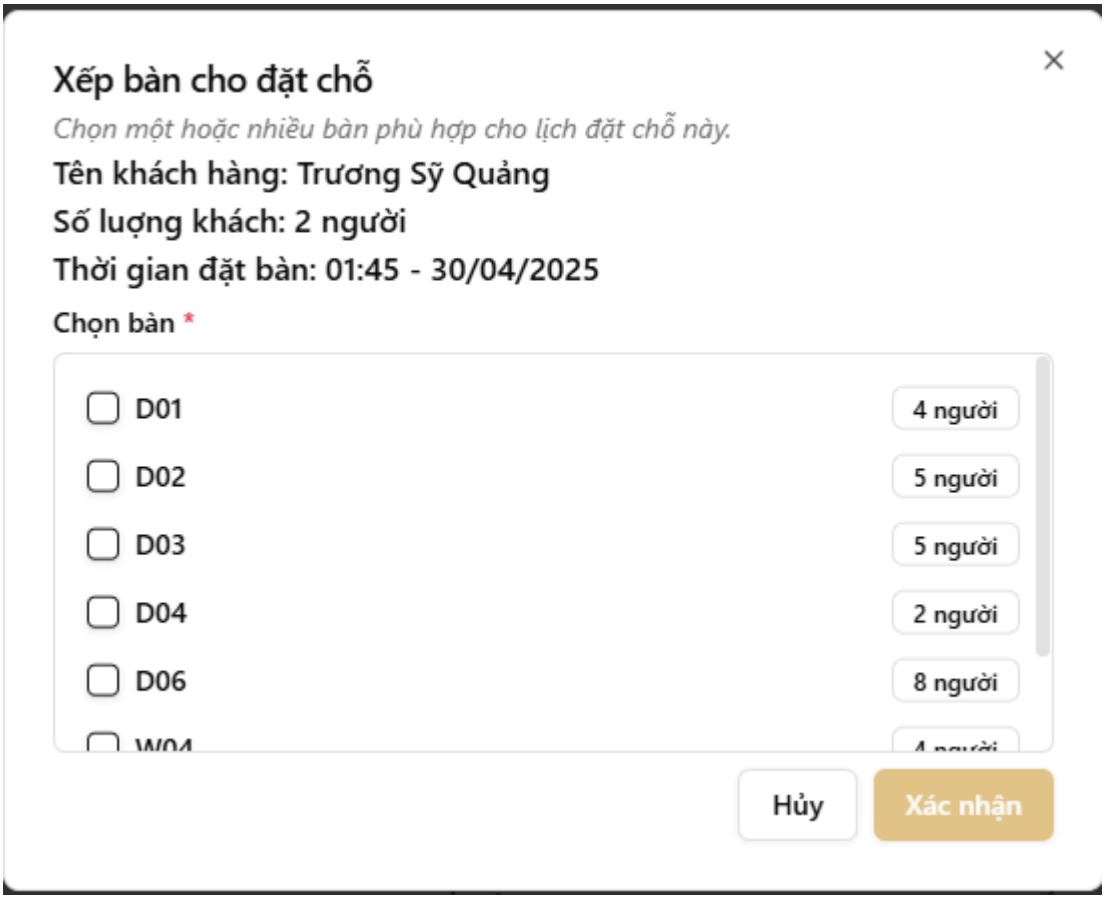
16. Manager checks availability & selects **“Xác nhận đặt bàn”** or **“Hủy đặt bàn”**

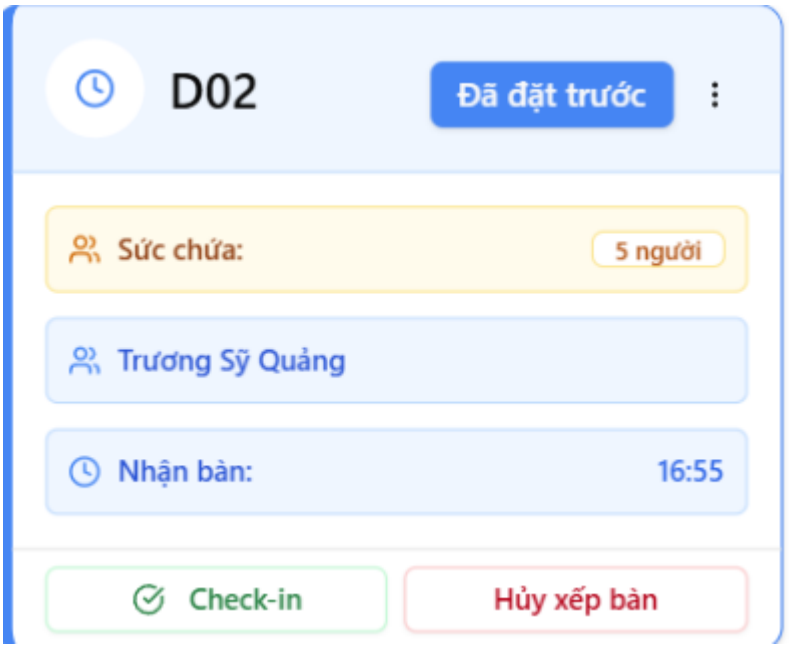


17. System sends reminder to Manager before booking time

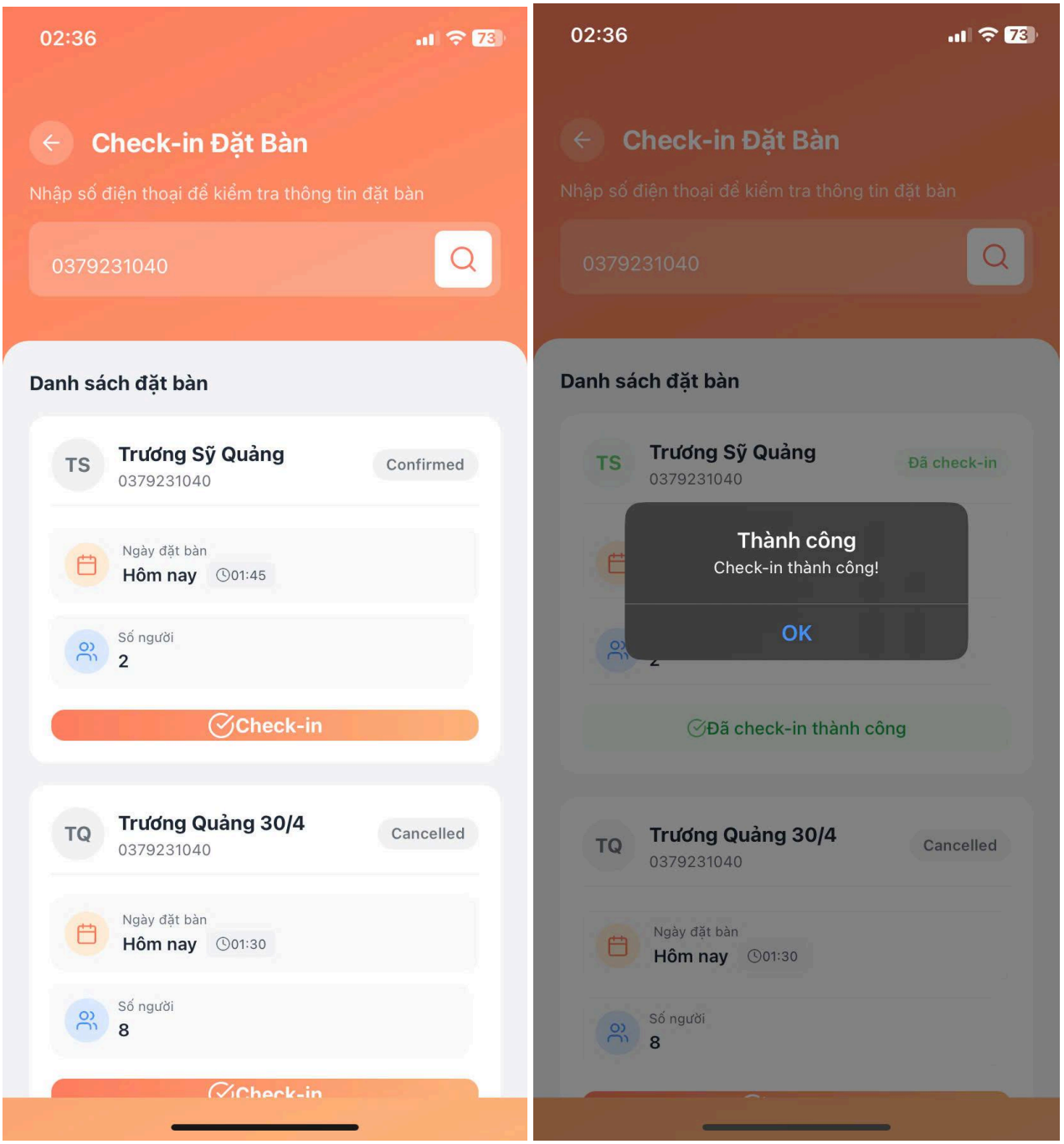


18. Manager assign table for booking





19. On arrival, staff searches phone number & taps **“Check-in”**

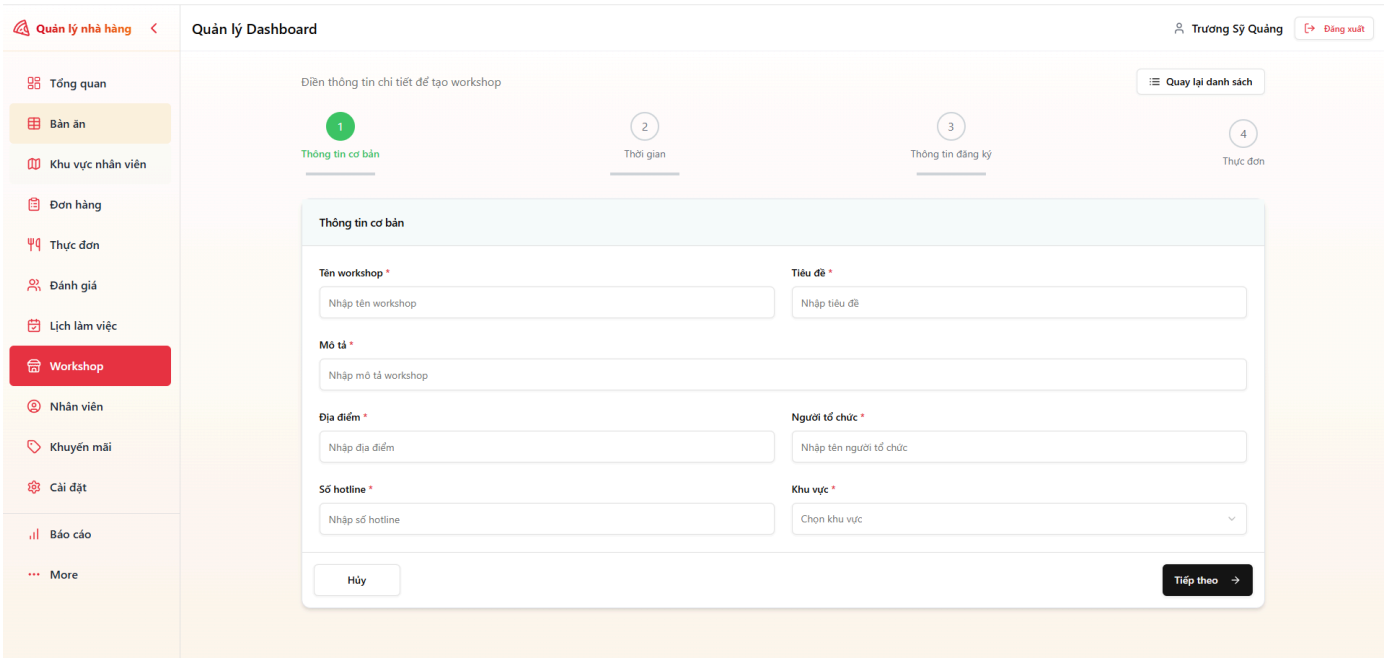


19. Reservation status becomes Checked-in

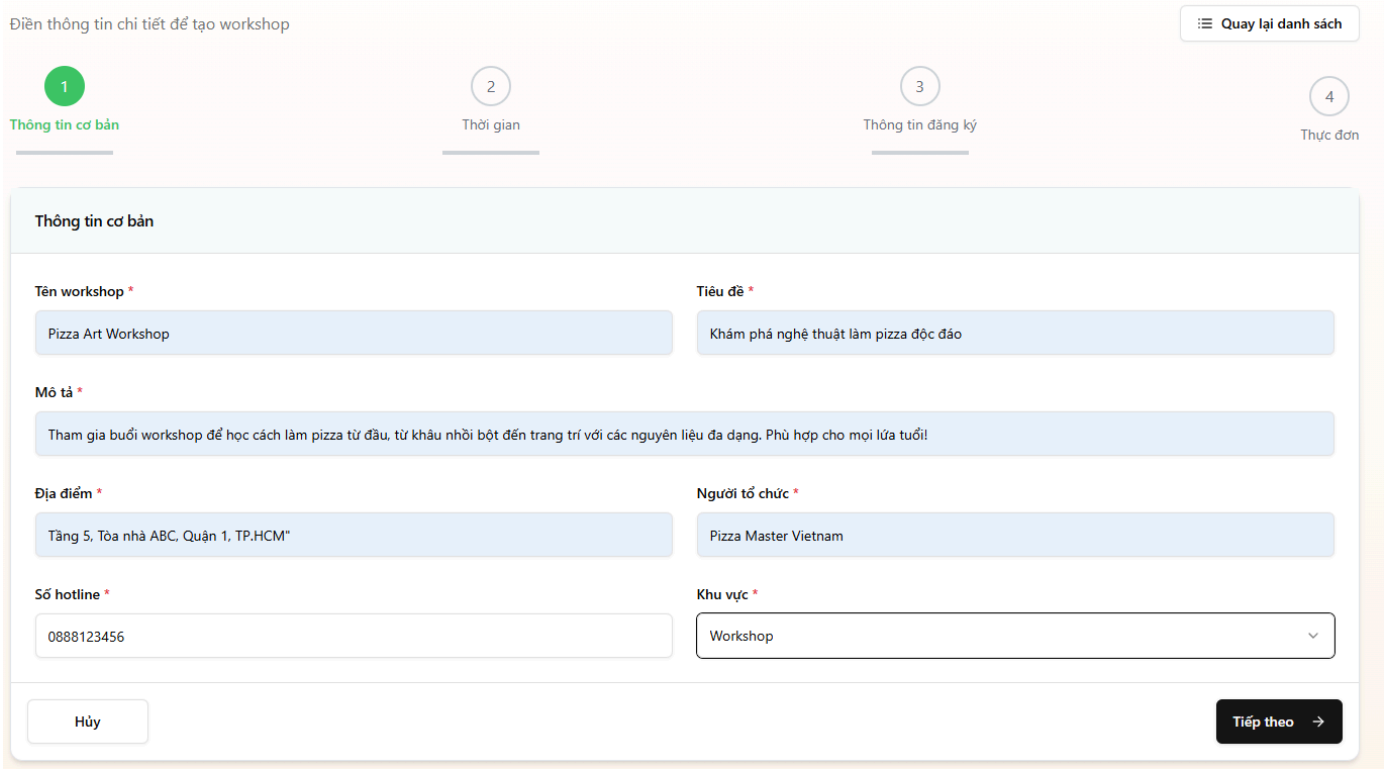
3.4 Workflow 3 – Pizza Workshop

Item	Description
Purpose	Enable Managers to create paid pizza-making workshops and let guests self-register.
Actors	Manager, Customer, Kitchen Staff, Waiter.
Prerequisites	Manager account with Workshop privileges. Workshop menu items configured.

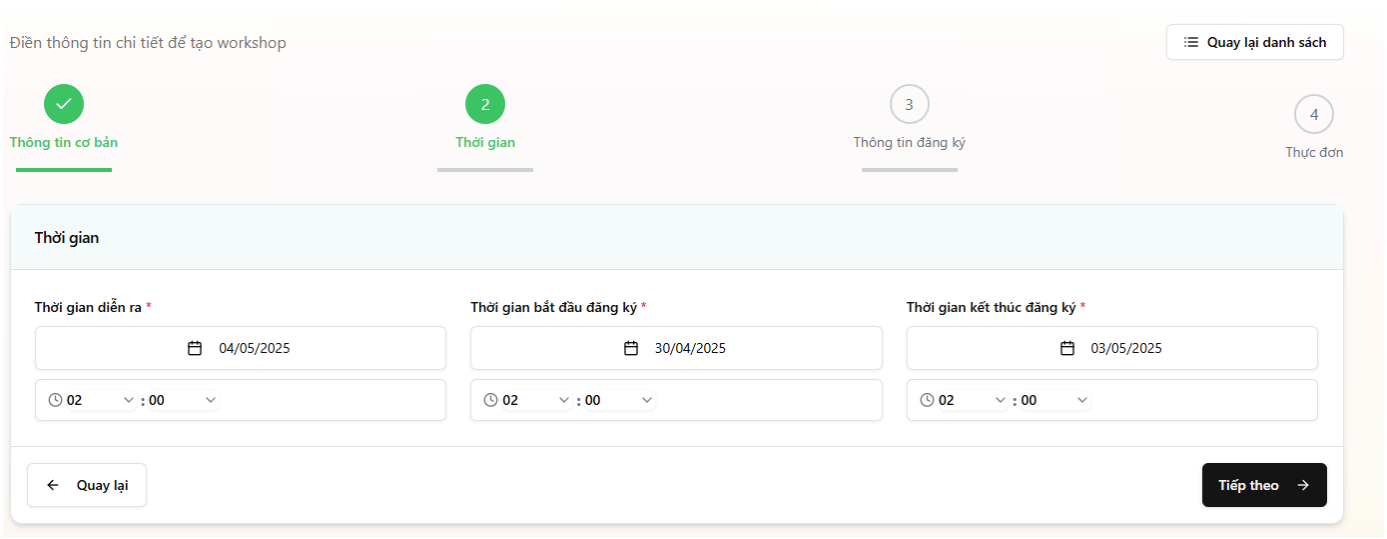
20. Manager → **“Workshop”** → **“Tạo Workshop”**



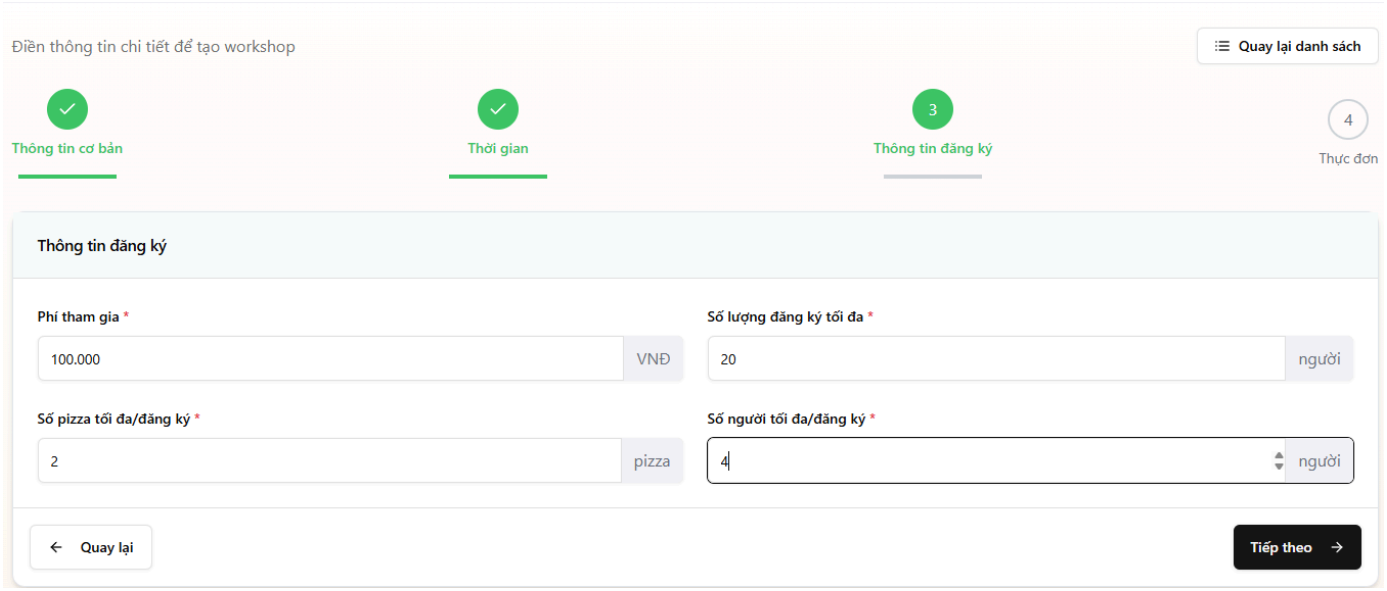
21. Step 1 – Enter Basic Info & click **“Tiếp theo”**



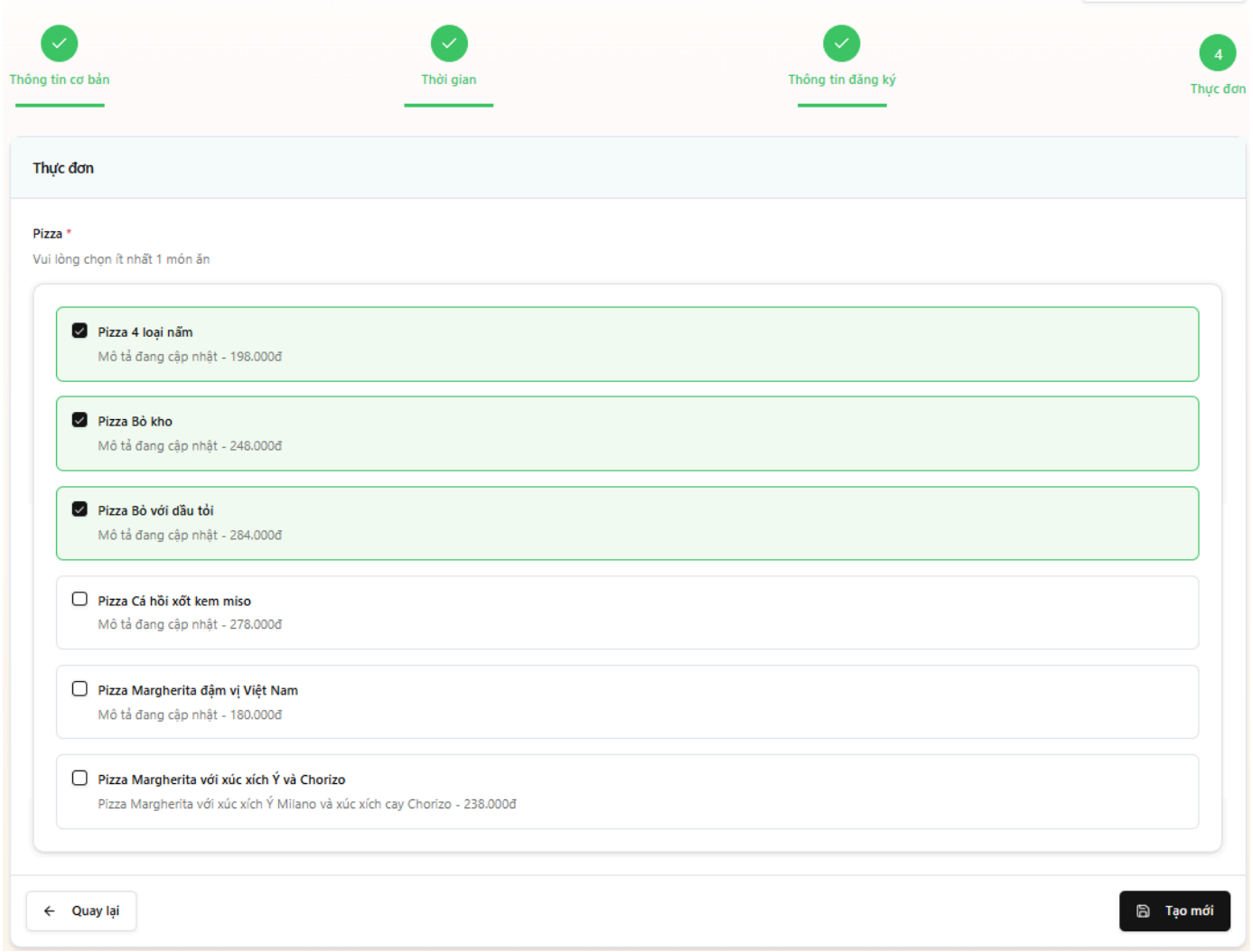
22. Step 2 – Set Schedule & click **“Tiếp theo”**



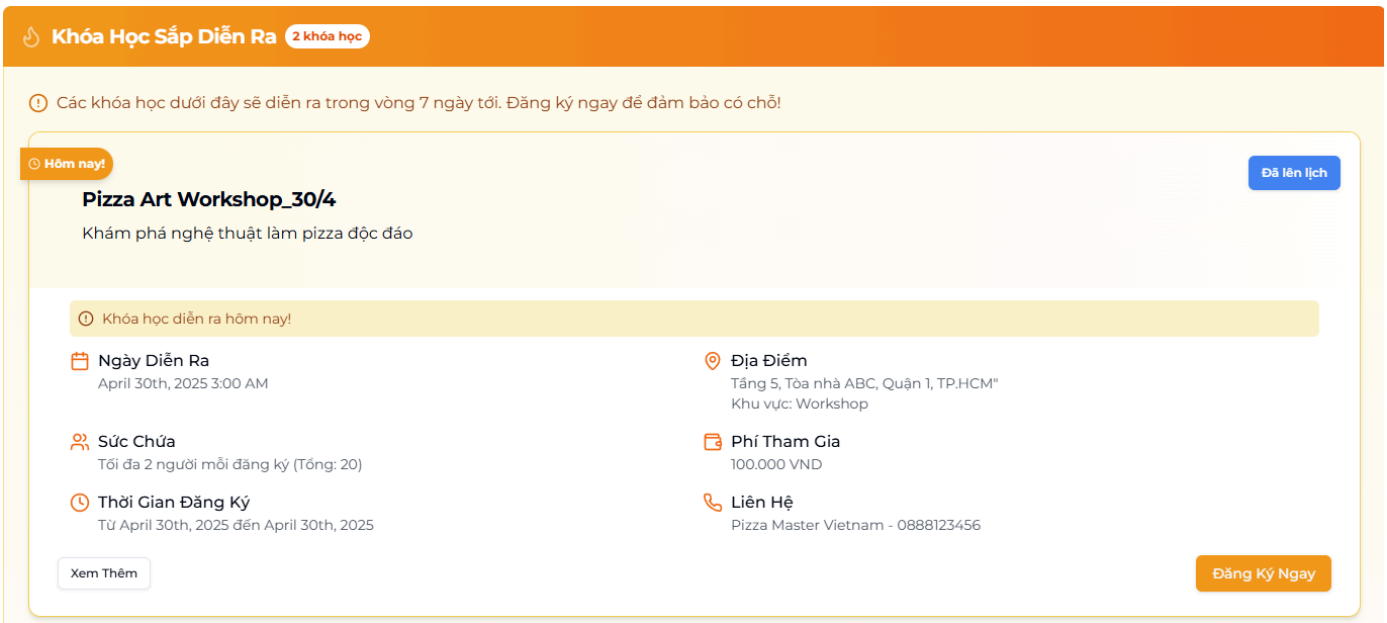
23. Step 3 – Configure Fee, Capacity & click **“Tiếp theo”**



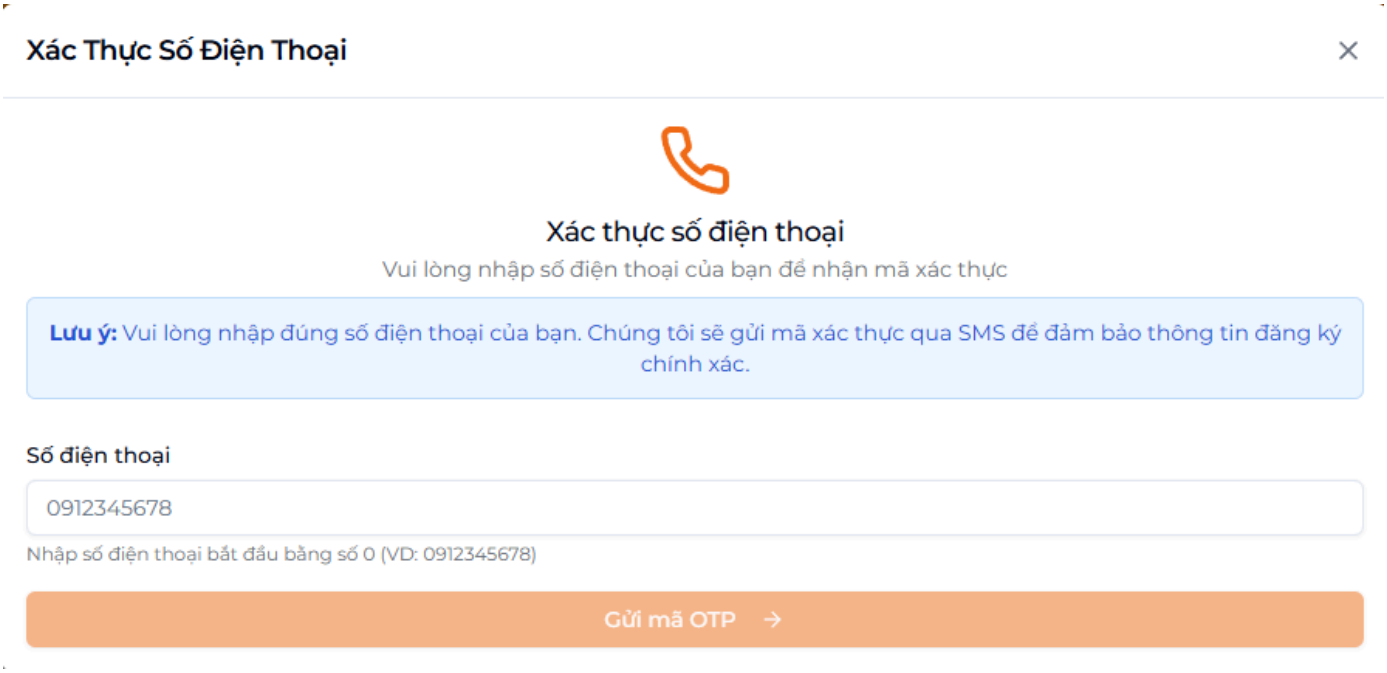
24. Step 4 – Choose at least 1 pizza & click **“Tạo mới”**

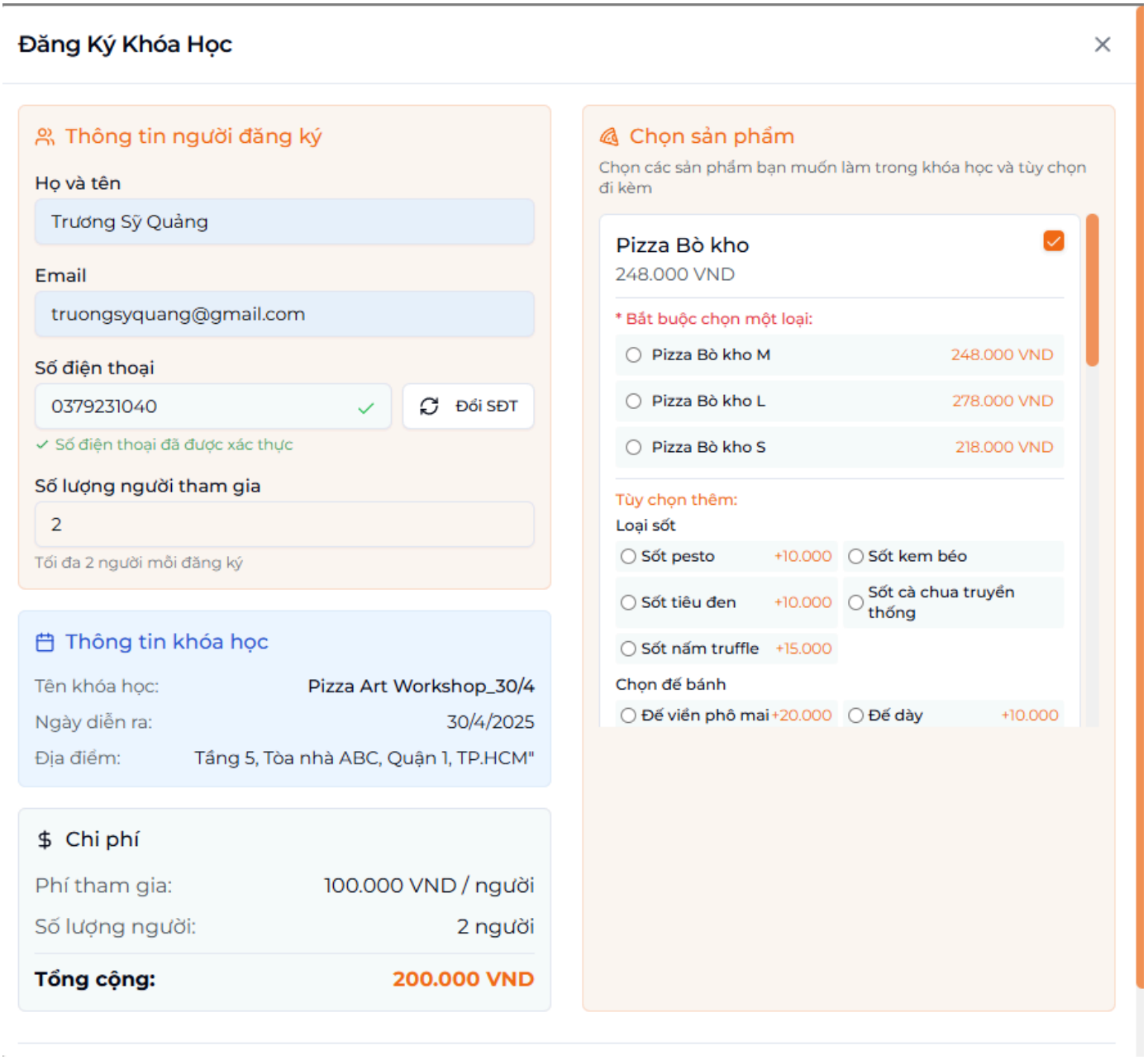


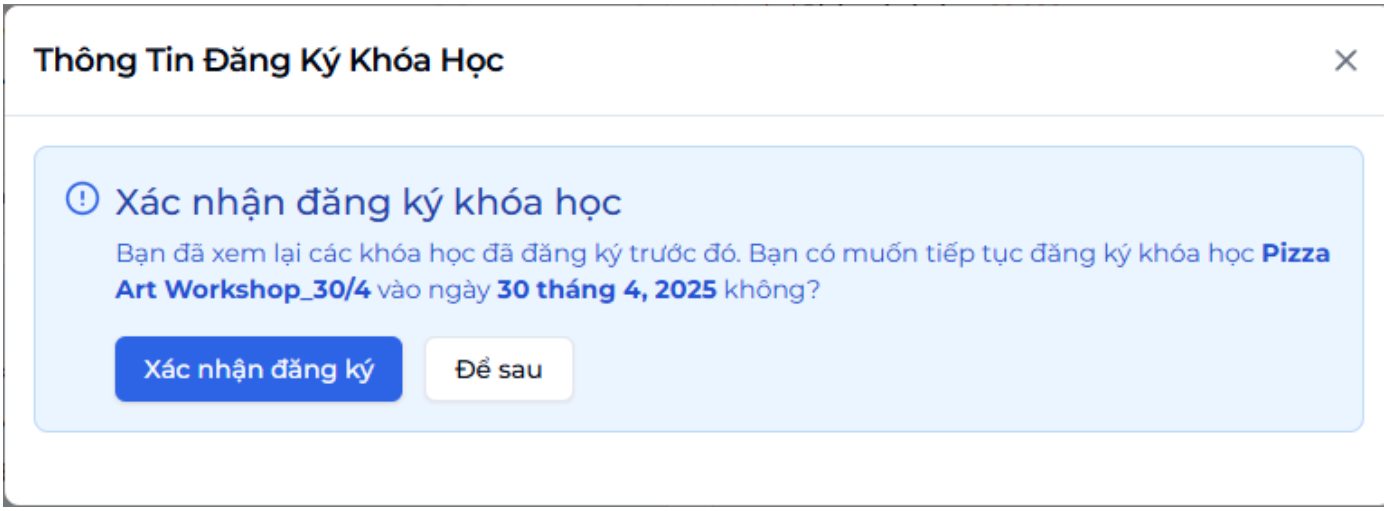
25. Customer → **Tab Workshop** → clicks **"Đăng ký ngay"**



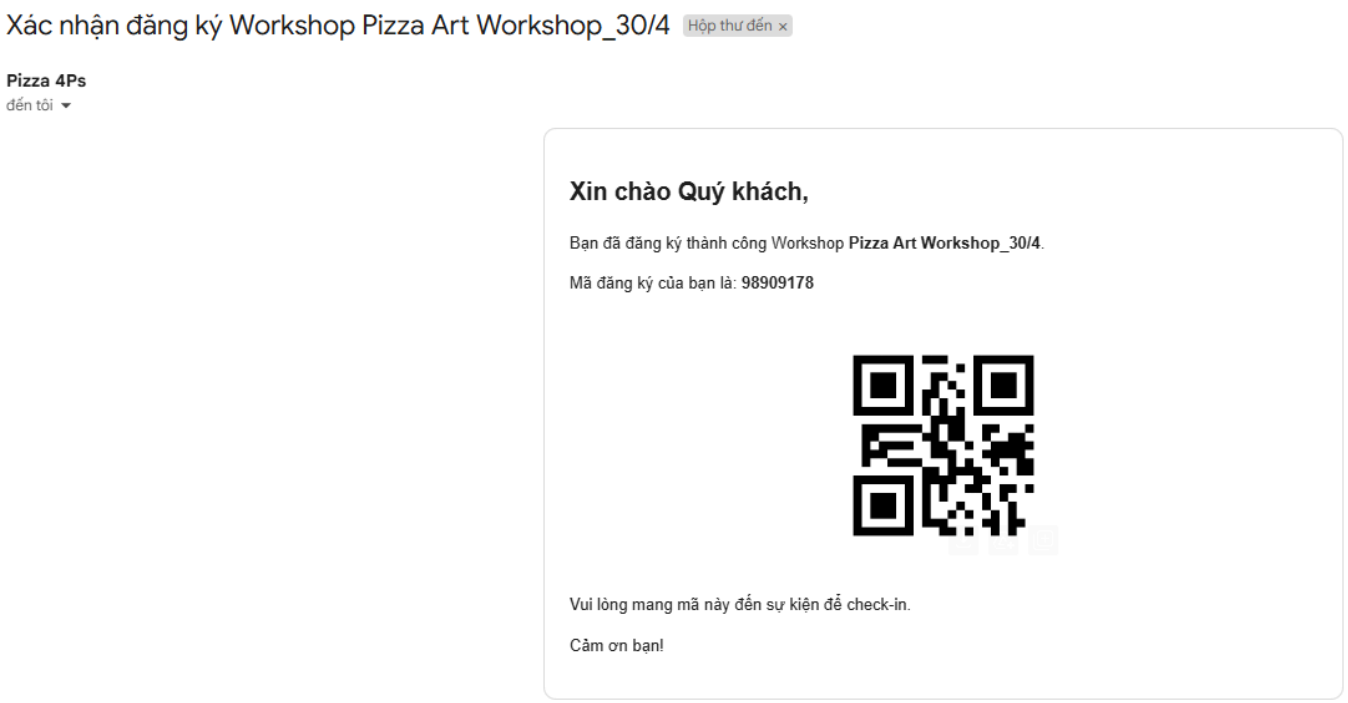
26. Customer enters phone, verifies OTP, and clicks **"Gửi mã OTP"**


Xác Thực Số Điện Thoại
Xác thực số điện thoại
Vui lòng nhập số điện thoại của bạn để nhận mã xác thực
Lưu ý: Vui lòng nhập đúng số điện thoại của bạn. Chúng tôi sẽ gửi mã xác thực qua SMS để đảm bảo thông tin đăng ký chính xác.
Số điện thoại
0912345678
Nhập số điện thoại bắt đầu bằng số 0 (VD: 0912345678)
Gửi mã OTP

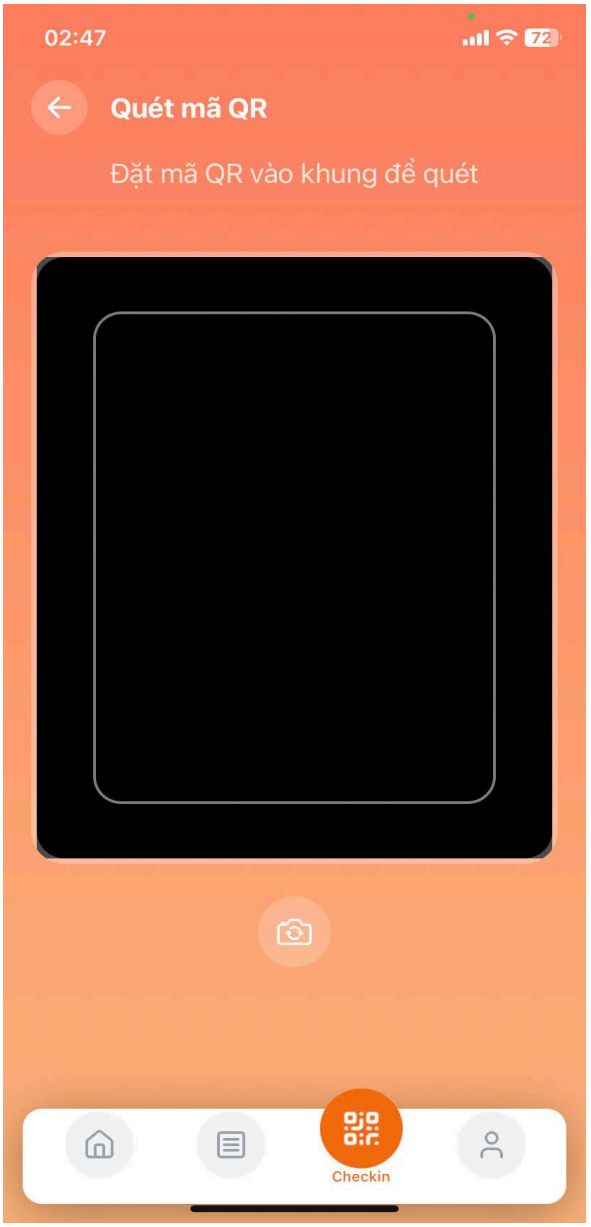

Đăng Ký Khóa Học
Thông tin người đăng ký
Họ và tên
Trương Sỹ Quảng
Email
truongsyquang@gmail.com
Số điện thoại
0379231040
Đổi SĐT
✓ Số điện thoại đã được xác thực
Số lượng người tham gia
2
Tối đa 2 người mỗi đăng ký
Thông tin khóa học
Tên khóa học: Pizza Art Workshop_30/4
Ngày diễn ra: 30/4/2025
Địa điểm: Tầng 5, Tòa nhà ABC, Quận 1, TP.HCM"
$ Chi phí
Phí tham gia: 100.000 VND / người
Số lượng người: 2 người
Tổng cộng: 200.000 VND
Chọn sản phẩm
Chọn các sản phẩm bạn muốn làm trong khóa học và tùy chọn đi kèm
Pizza Bò kho
248.000 VND
* Bắt buộc chọn một loại:
Pizza Bò kho M 248.000 VND
Pizza Bò kho L 278.000 VND
Pizza Bò kho S 218.000 VND
Tùy chọn thêm:
Loại sốt
Sốt pesto +10.000
Sốt kem béo
Sốt tiêu đen +10.000
Sốt cà chua truyền thống
Sốt nấm truffle +15.000
Chọn đế bánh
Đế viền phô mai +20.000
Đế dày +10.000

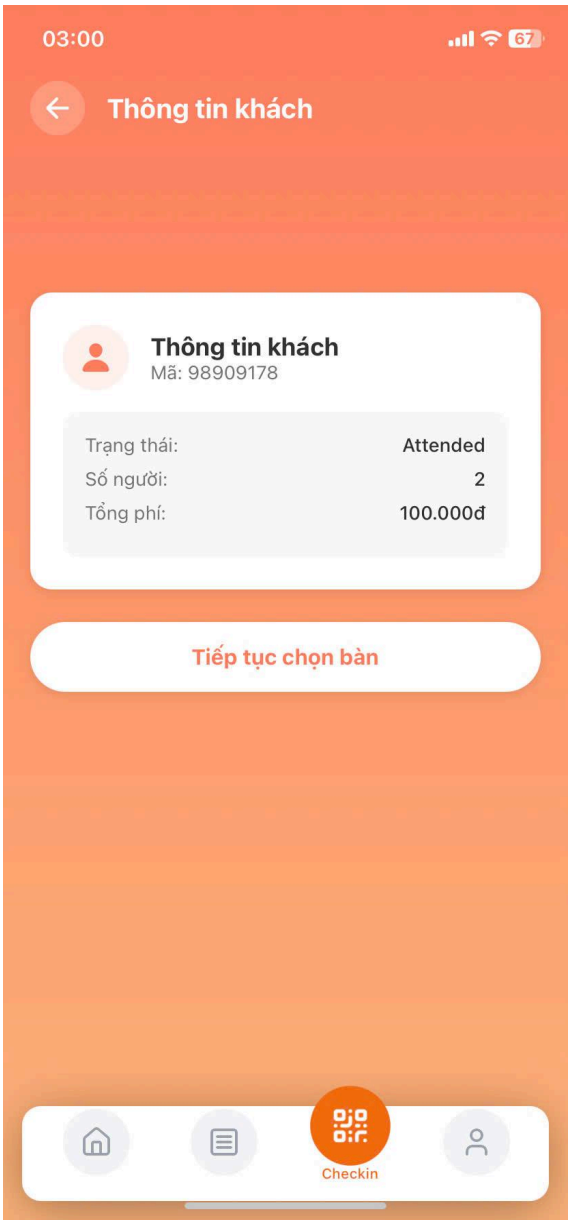

Thông Tin Đăng Ký Khóa Học
Xác nhận đăng ký khóa học
Bạn đã xem lại các khóa học đã đăng ký trước đó. Bạn có muốn tiếp tục đăng ký khóa học Pizza Art Workshop_30/4 vào ngày 30 tháng 4, 2025 không?
Xác nhận đăng ký
Để sau

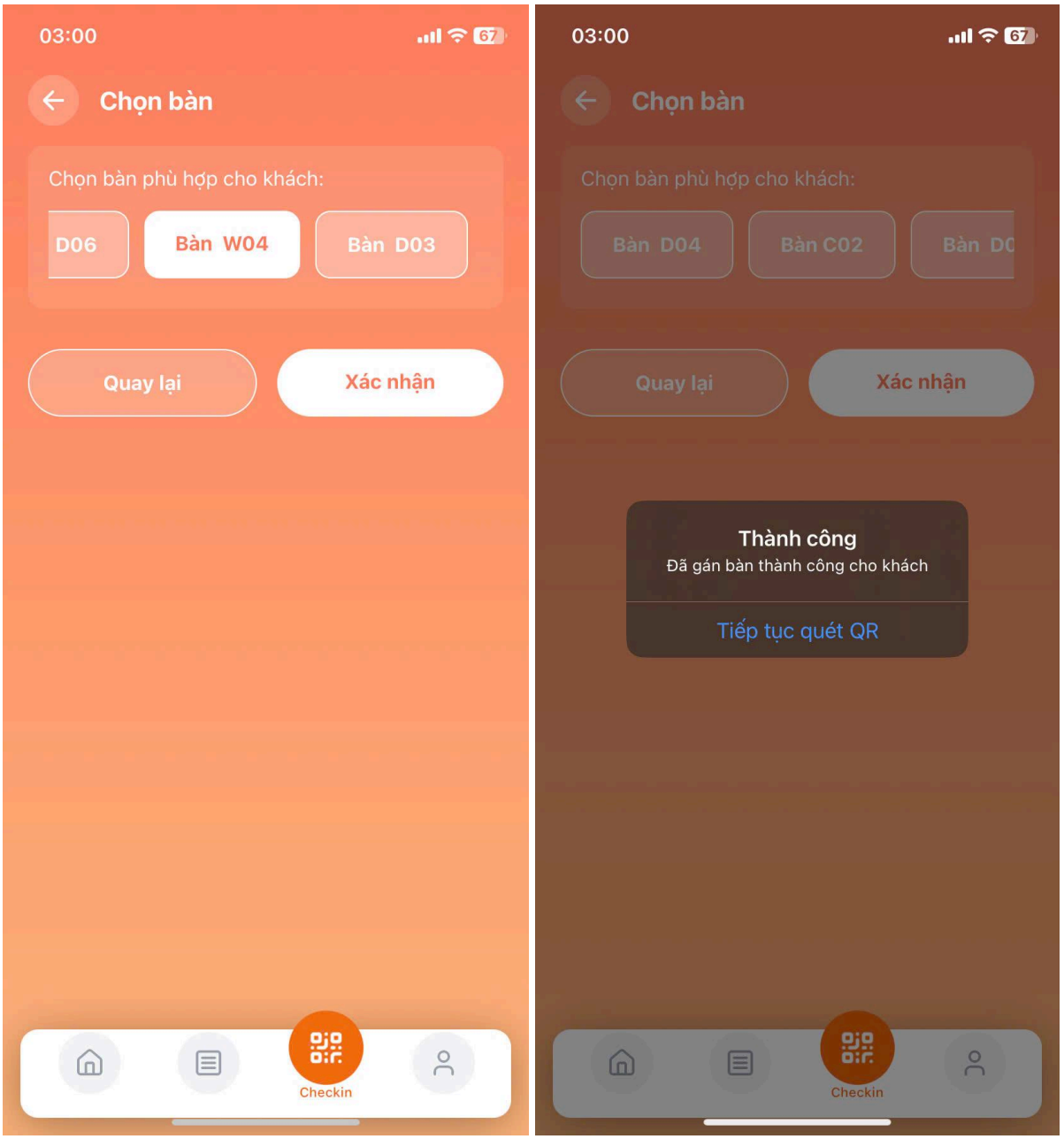
27. System emails QR-ticket to customer



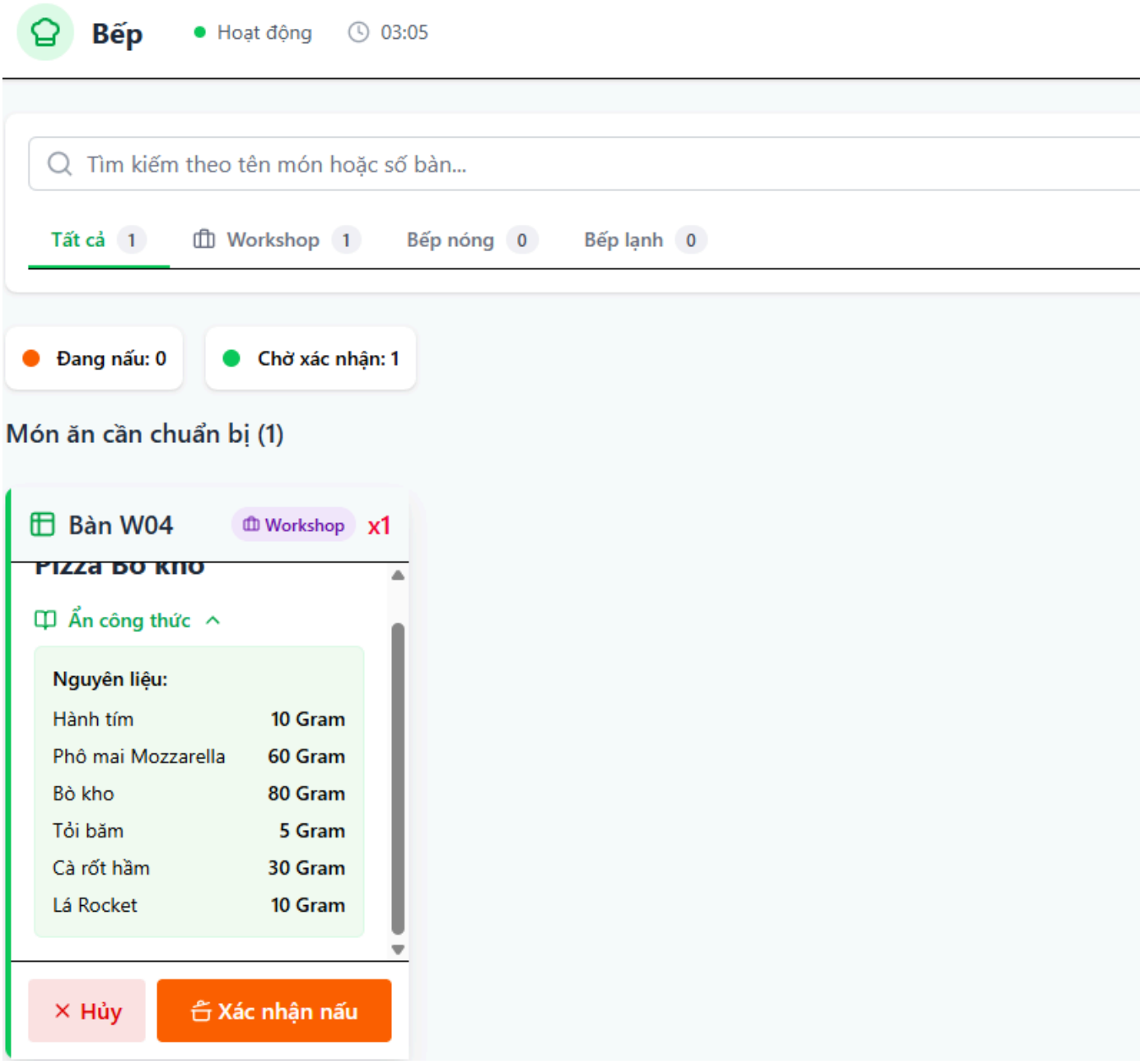
28. Workshop day – Staff scans QR and taps **“Tiếp tục chọn bàn”**

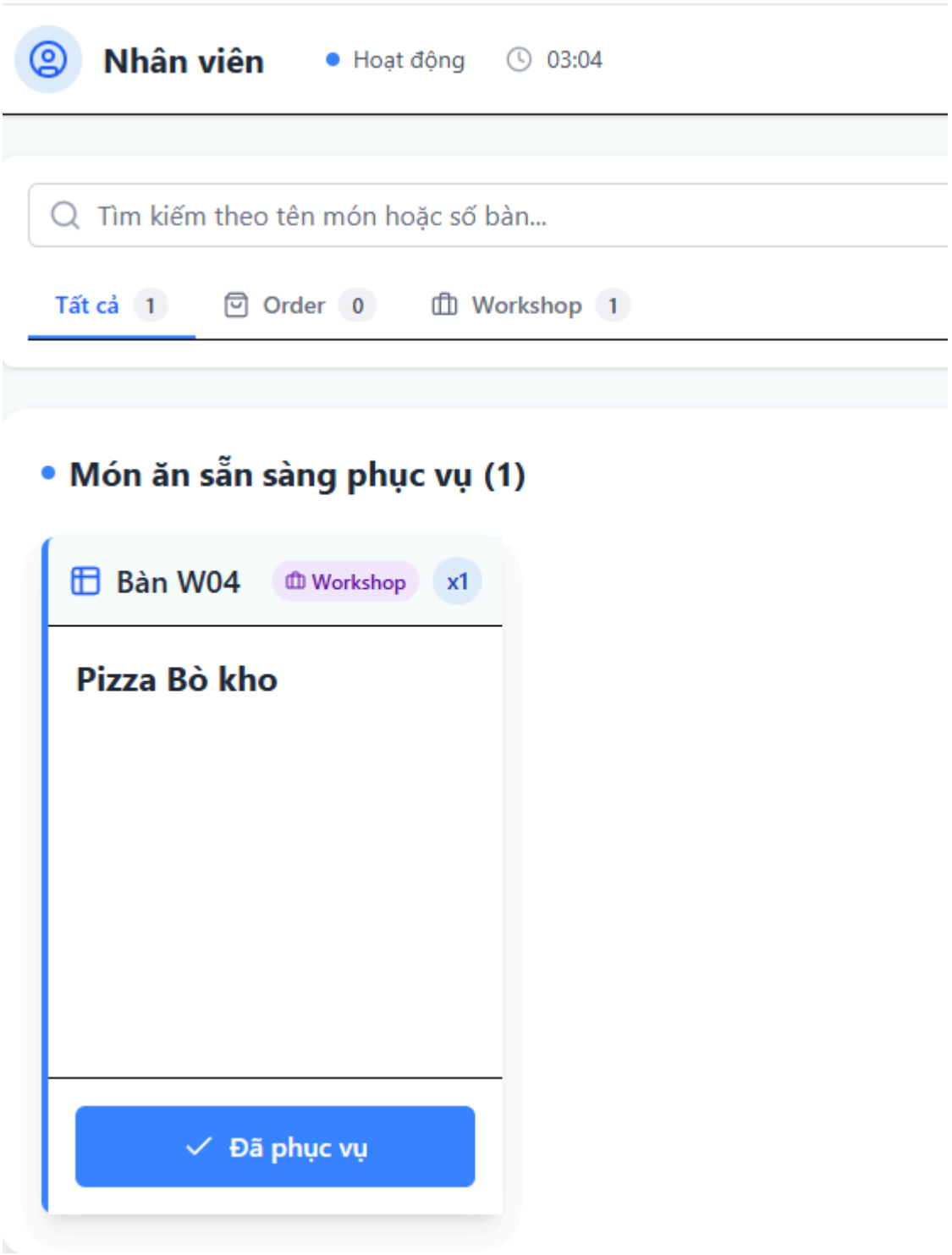

02:47
Quét mã QR
Đặt mã QR vào khung để quét
Checkin


03:00
Thông tin khách
Thông tin khách
Mã: 98909178
Trạng thái: Attended
Số người: 2
Tổng phí: 100.000đ
Tiếp tục chọn bàn
Checkin

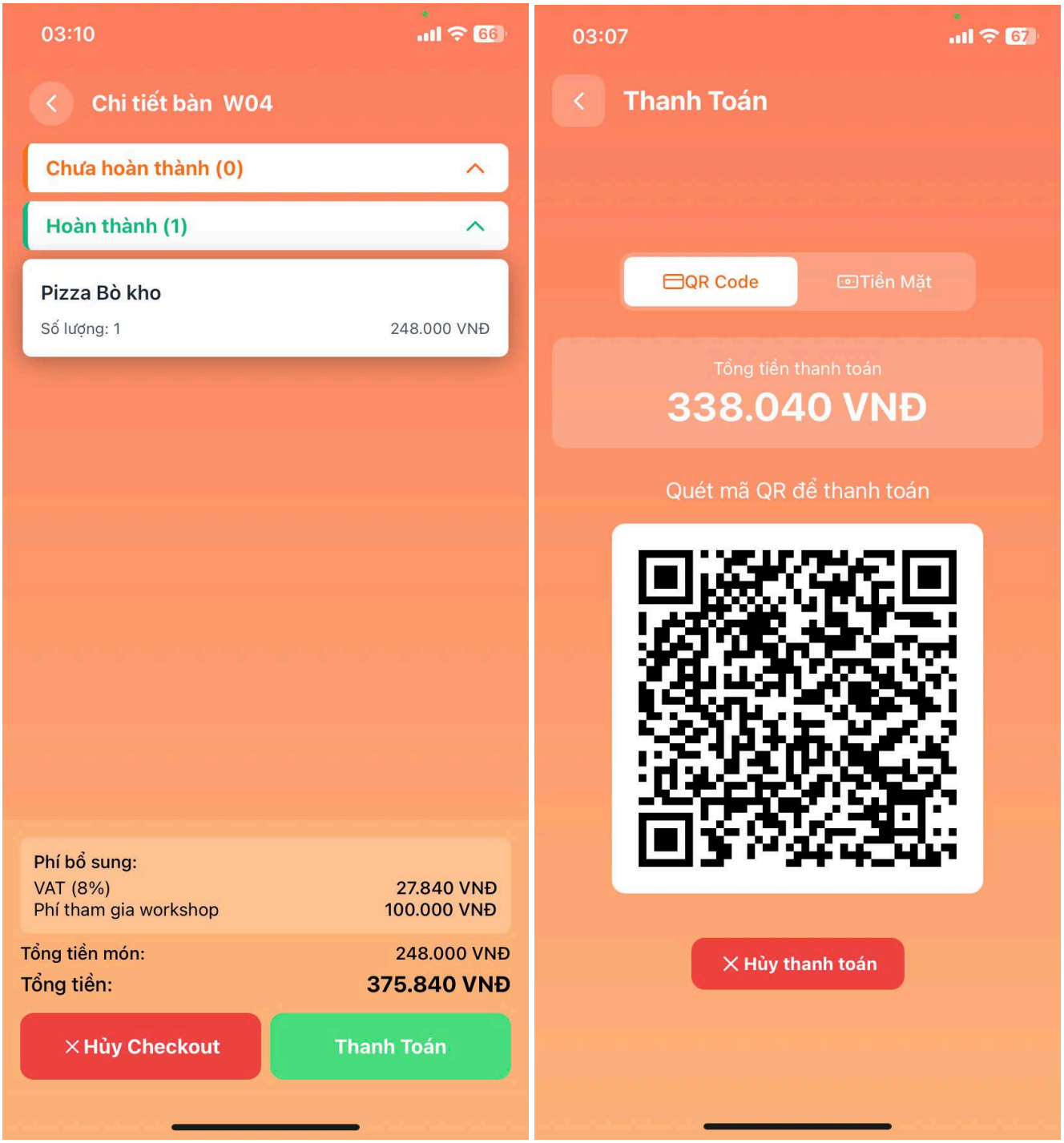


29. Kitchen prepares ingredients ⇒ Waiter serves


Bếp
Hoạt động
03:05
Tìm kiếm theo tên món hoặc số bàn...
Tất cả 1
Workshop 1
Bếp nóng 0
Bếp lạnh 0
Đang nấu: 0
Chờ xác nhận: 1
Món ăn cần chuẩn bị (1)
Bàn W04
Workshop
x1
Ẩn công thức
Nguyên liệu:
Hành tím 10 Gram
Phô mai Mozzarella 60 Gram
Bò kho 80 Gram
Tỏi băm 5 Gram
Cà rốt hầm 30 Gram
Lá Rocket 10 Gram
Hủy
Xác nhận nấu



30. Checkout & payment completed; table Closed

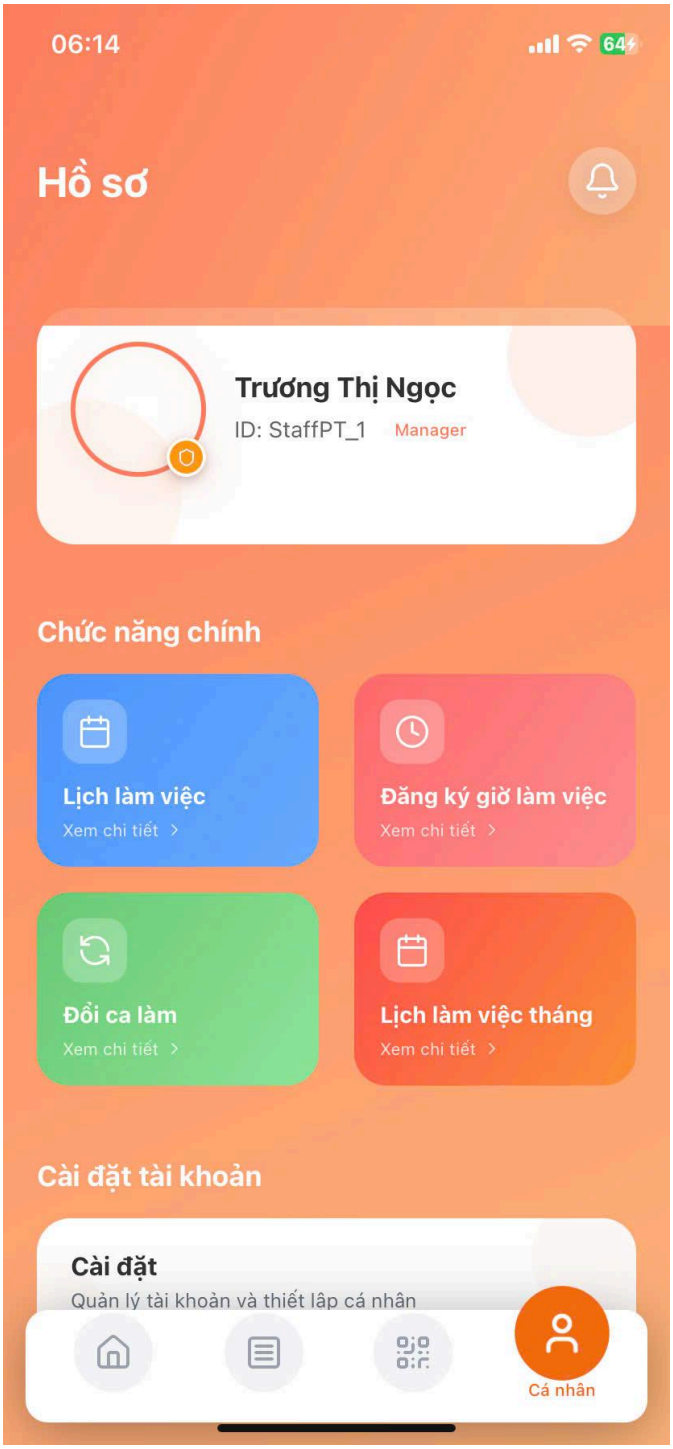


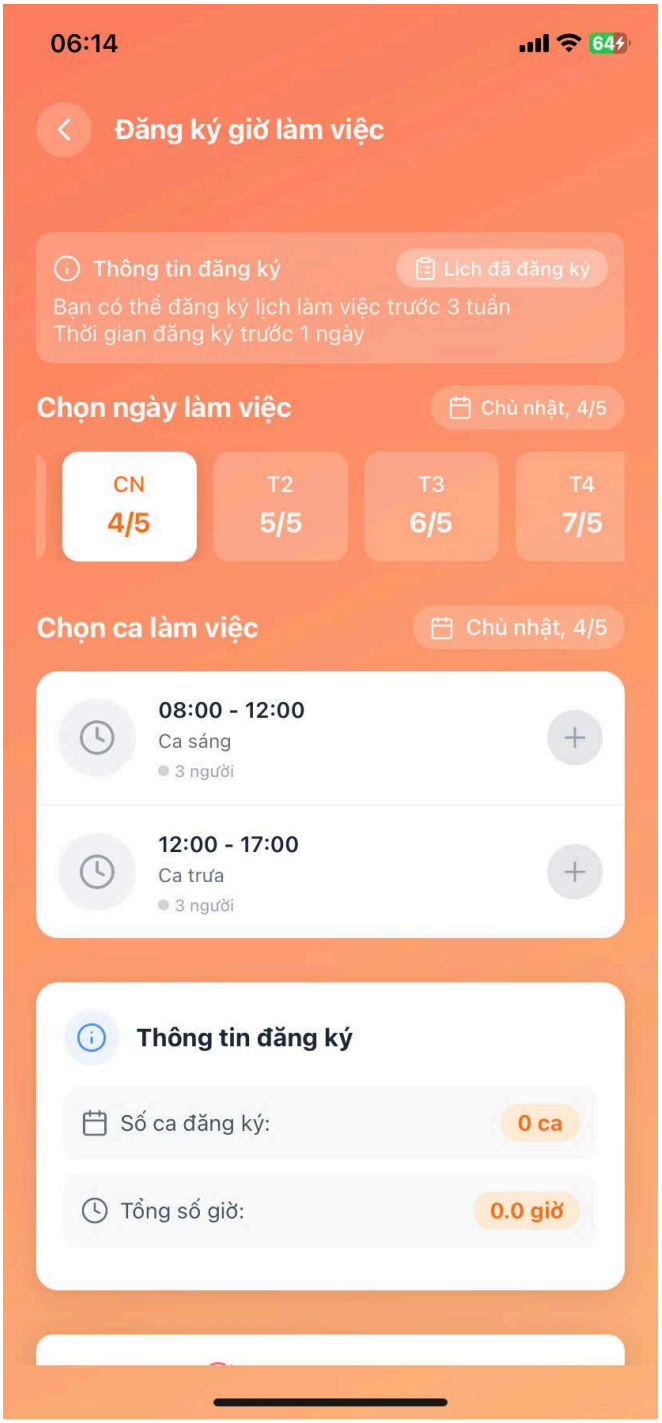
3.5 Workflow 4 – Part-Time Shift Registration

Item	Description

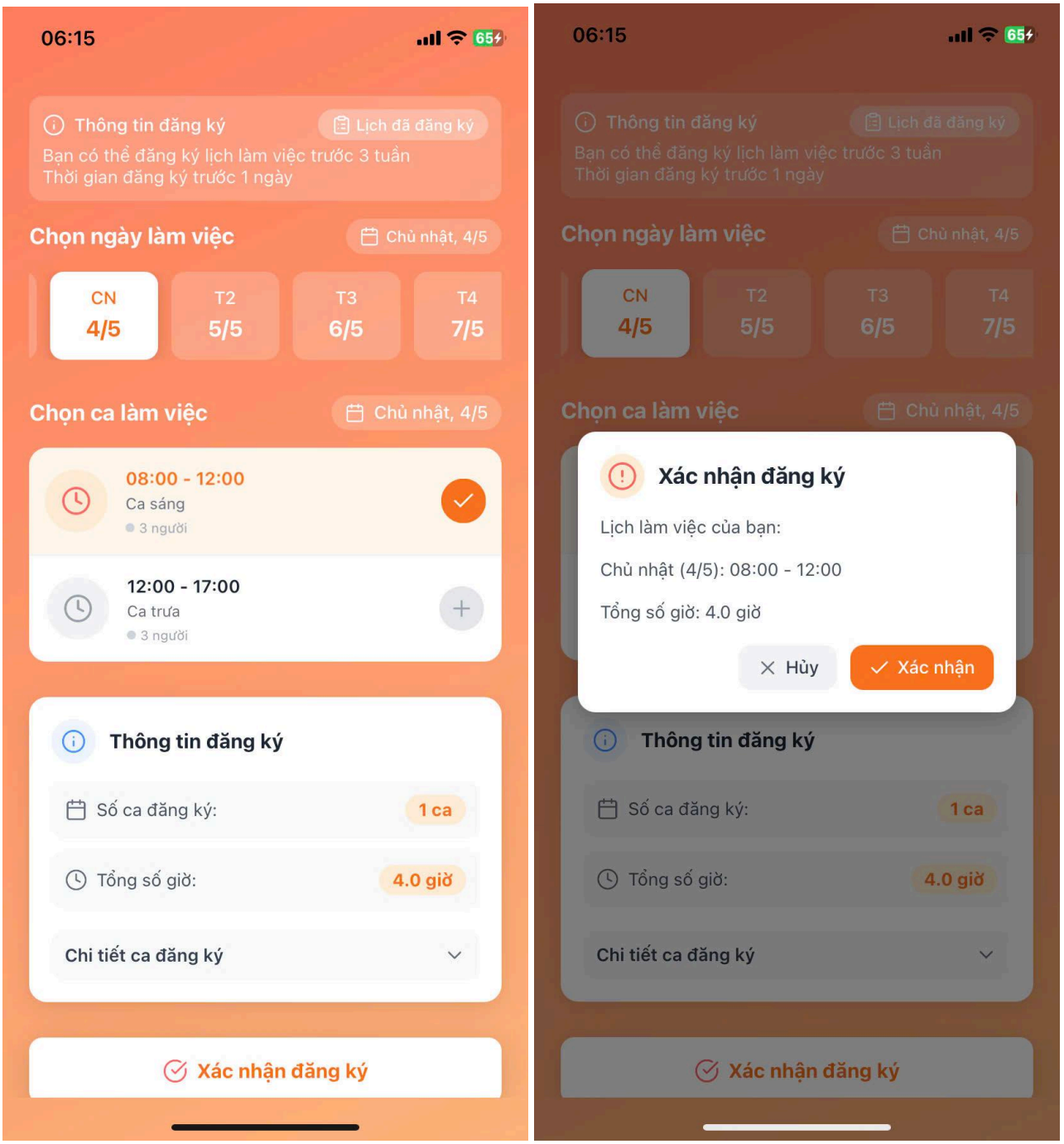
Purpose	Allow part-time employees to register working shifts and for Managers to approve and assign areas.
Actors	Part-Time Employee, Manager.
Prerequisites	Employee account with role Part-time. Shift templates configured.

31. Employee goes to **“Cá nhân”** → **“Đăng ký ca làm”**

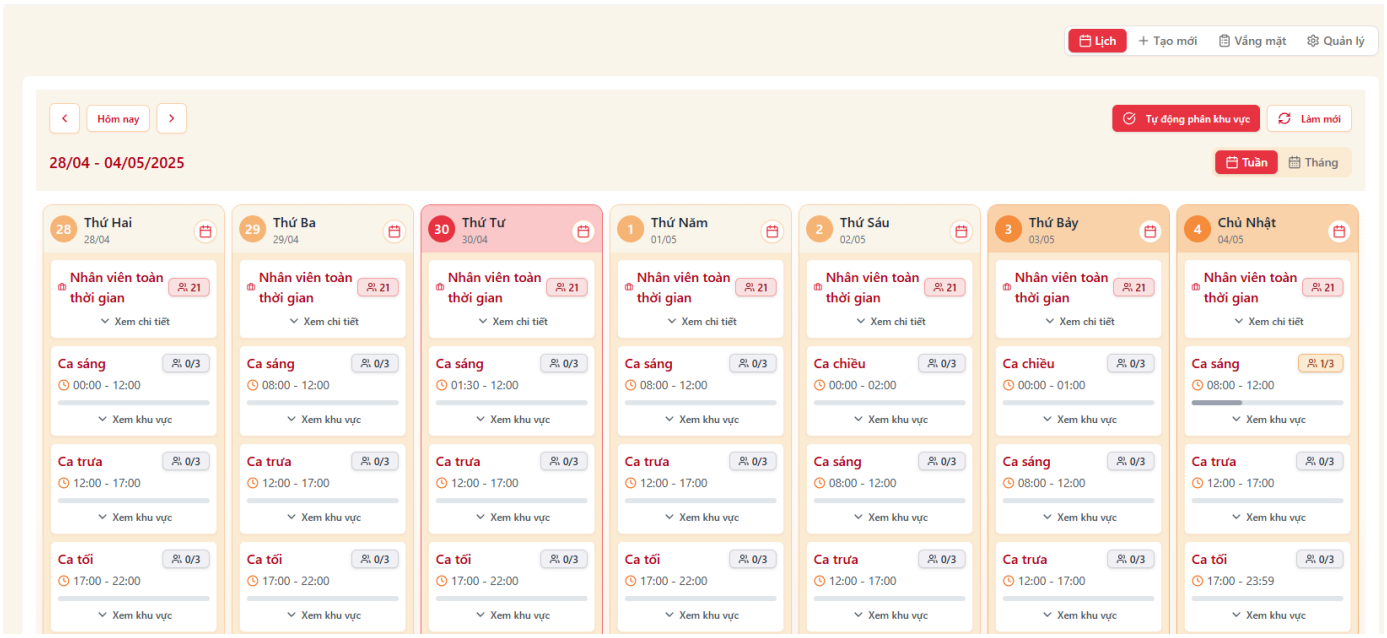


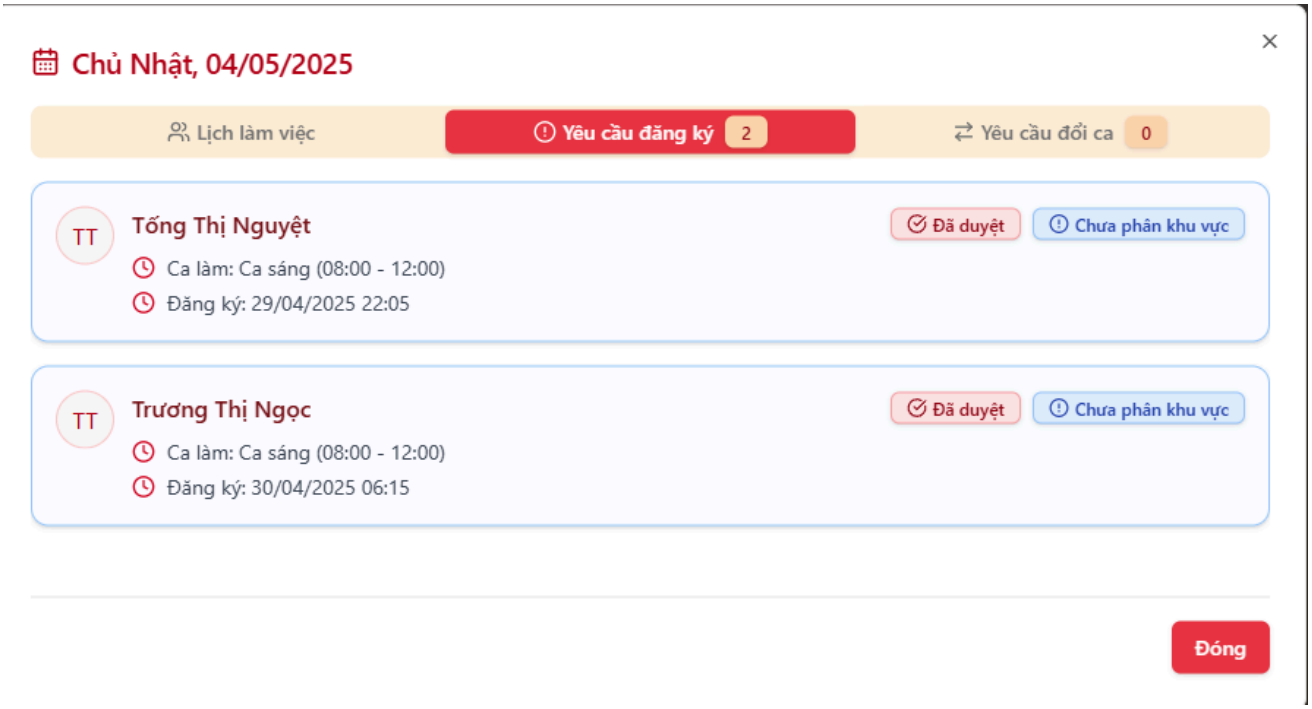


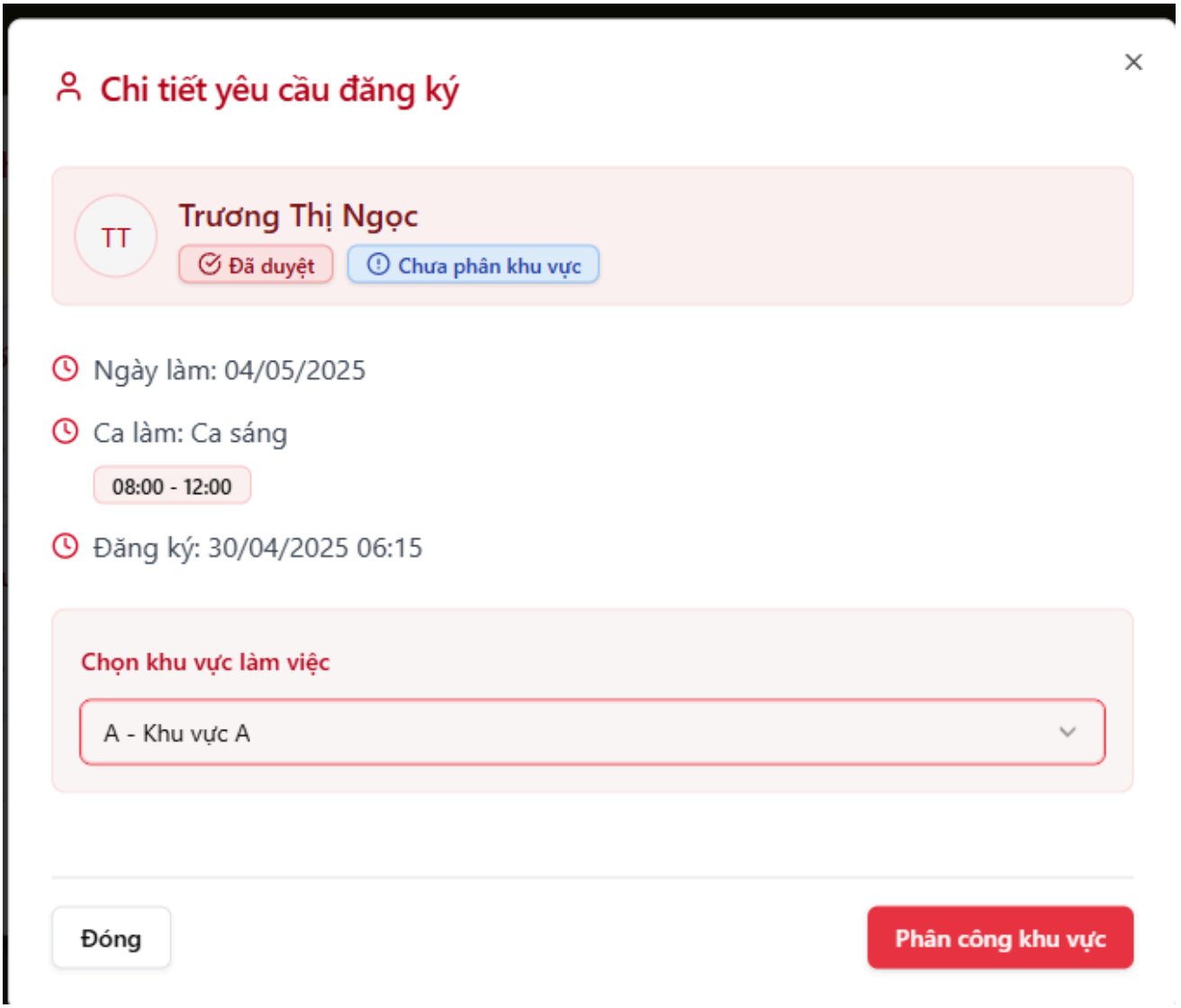
32. Employee selects shift & taps **“Xác nhận đăng ký”** → Status: *Approved* or *On Hold*



33. Manager → **“Lịch làm việc”** → either manually approves or auto-allocates working area

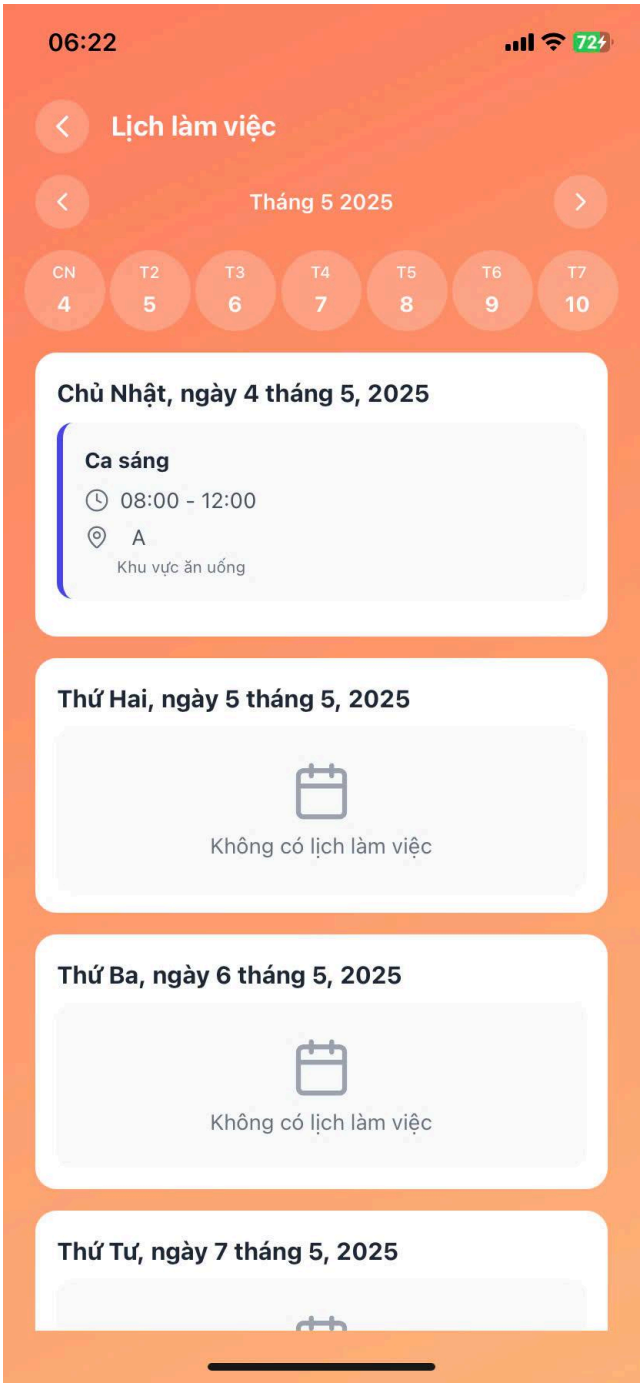

Lịch
Tạo mới
Vắng mặt
Quản lý
Hôm nay
Tự động phân khu vực
Làm mới
28/04 - 04/05/2025
Tuần
Tháng
28 Thứ Hai 28/04
29 Thứ Ba 29/04
30 Thứ Tư 30/04
1 Thứ Năm 01/05
2 Thứ Sáu 02/05
3 Thứ Bảy 03/05
4 Chủ Nhật 04/05
Nhân viên toàn thời gian 21
Xem chi tiết
Ca sáng 0/3 00:00 - 12:00
Ca sáng 0/3 08:00 - 12:00
Ca sáng 0/3 01:30 - 12:00
Ca sáng 0/3 08:00 - 12:00
Ca chiều 0/3 00:00 - 02:00
Ca chiều 0/3 00:00 - 01:00
Ca sáng 1/3 08:00 - 12:00
Ca trưa 0/3 12:00 - 17:00
Ca tối 0/3 17:00 - 22:00
Ca tối 0/3 17:00 - 23:59
Xem khu vực


Chủ Nhật, 04/05/2025
Lịch làm việc
Yêu cầu đăng ký 2
Yêu cầu đổi ca 0
TT
Tống Thị Nguyệt
Đã duyệt
Chưa phân khu vực
Ca làm: Ca sáng (08:00 - 12:00)
Đăng ký: 29/04/2025 22:05
TT
Trương Thị Ngọc
Đã duyệt
Chưa phân khu vực
Ca làm: Ca sáng (08:00 - 12:00)
Đăng ký: 30/04/2025 06:15
Đóng

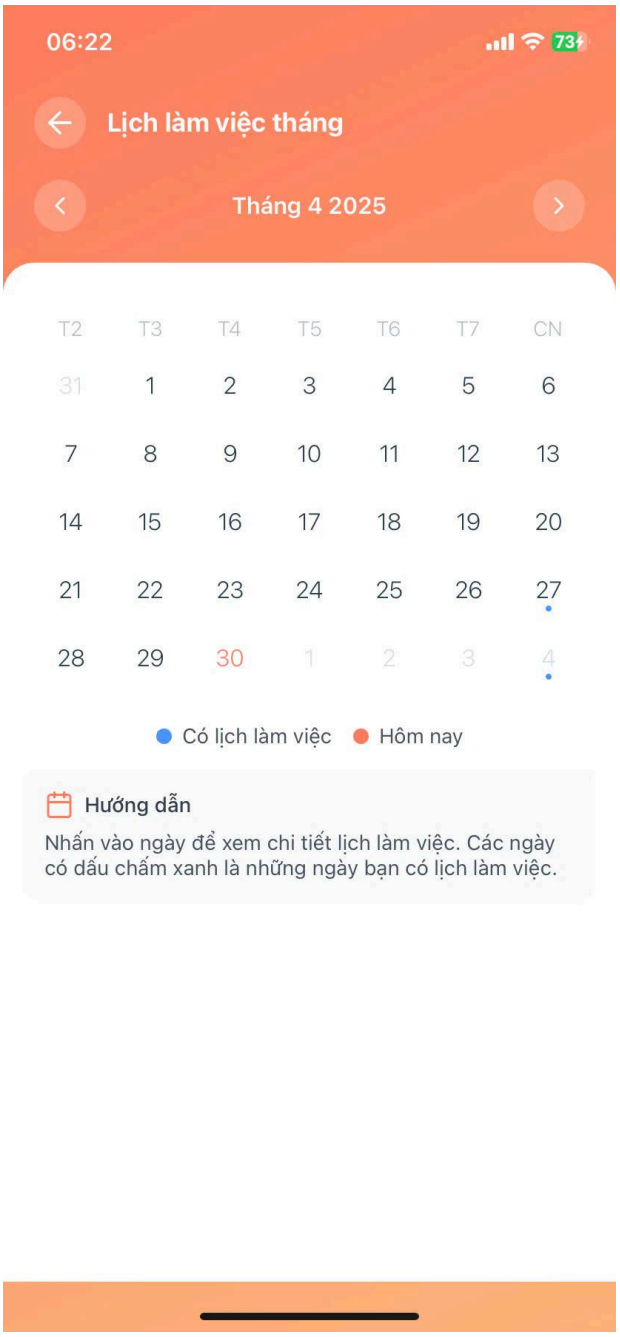

Chi tiết yêu cầu đăng ký
TT
Trương Thị Ngọc
Đã duyệt
Chưa phân khu vực
Ngày làm: 04/05/2025
Ca làm: Ca sáng
08:00 - 12:00
Đăng ký: 30/04/2025 06:15
Chọn khu vực làm việc
A - Khu vực A
Đóng
Phân công khu vực

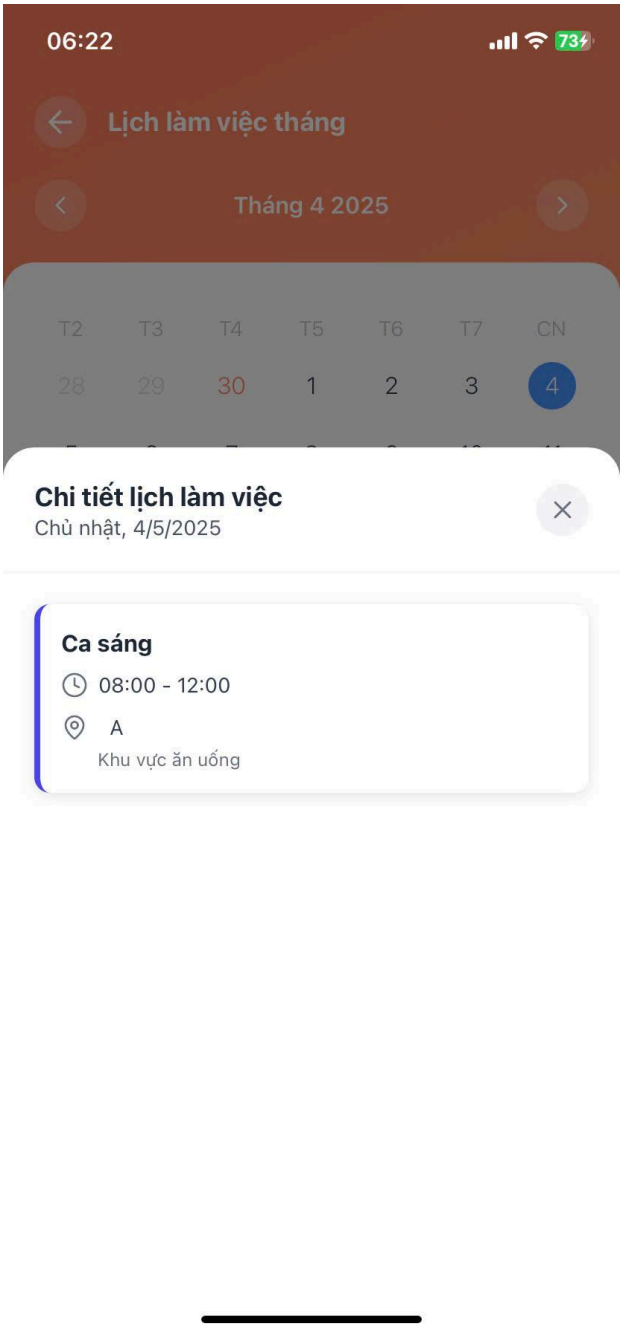
33. Staff view work schedules on the app

- View Weekly



- View by month



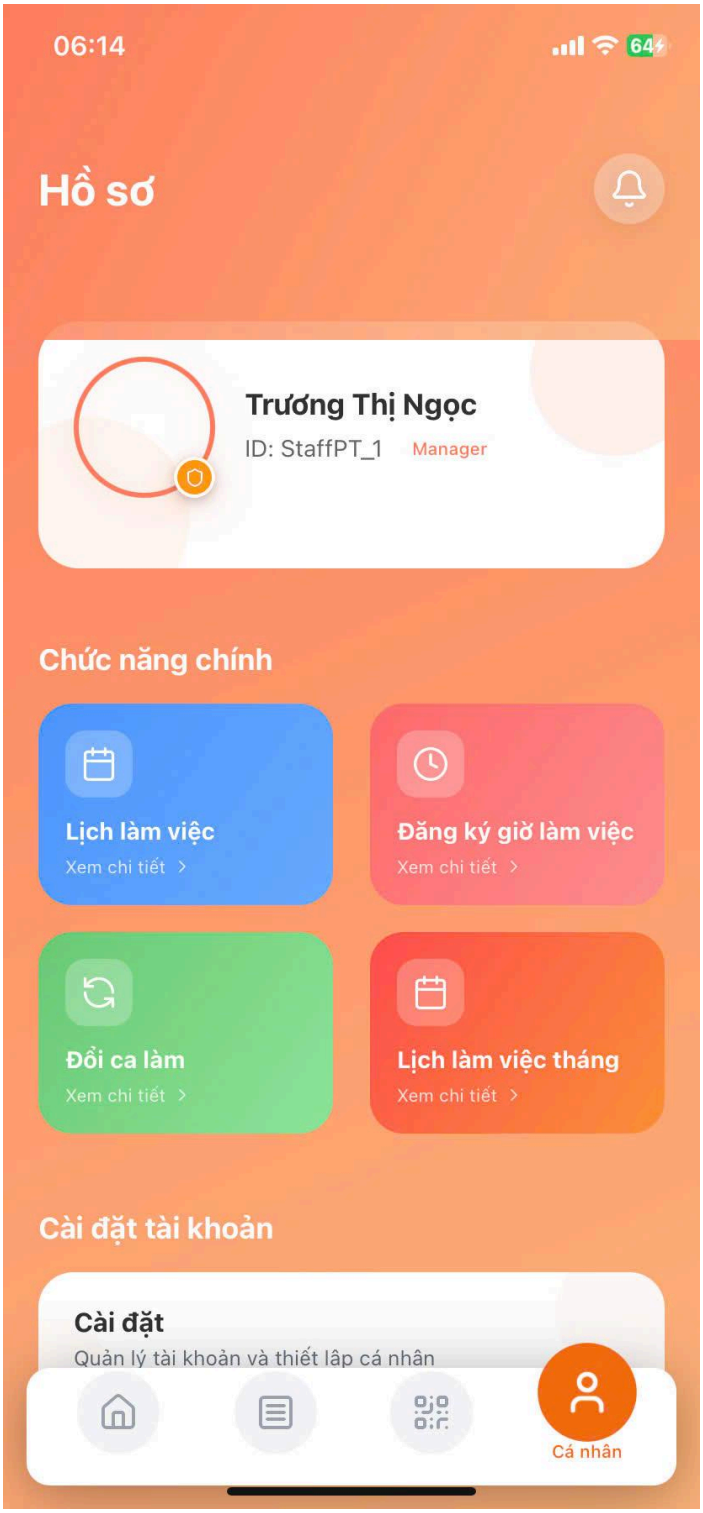


3.6 Workflow 5 – Shift Swap Request

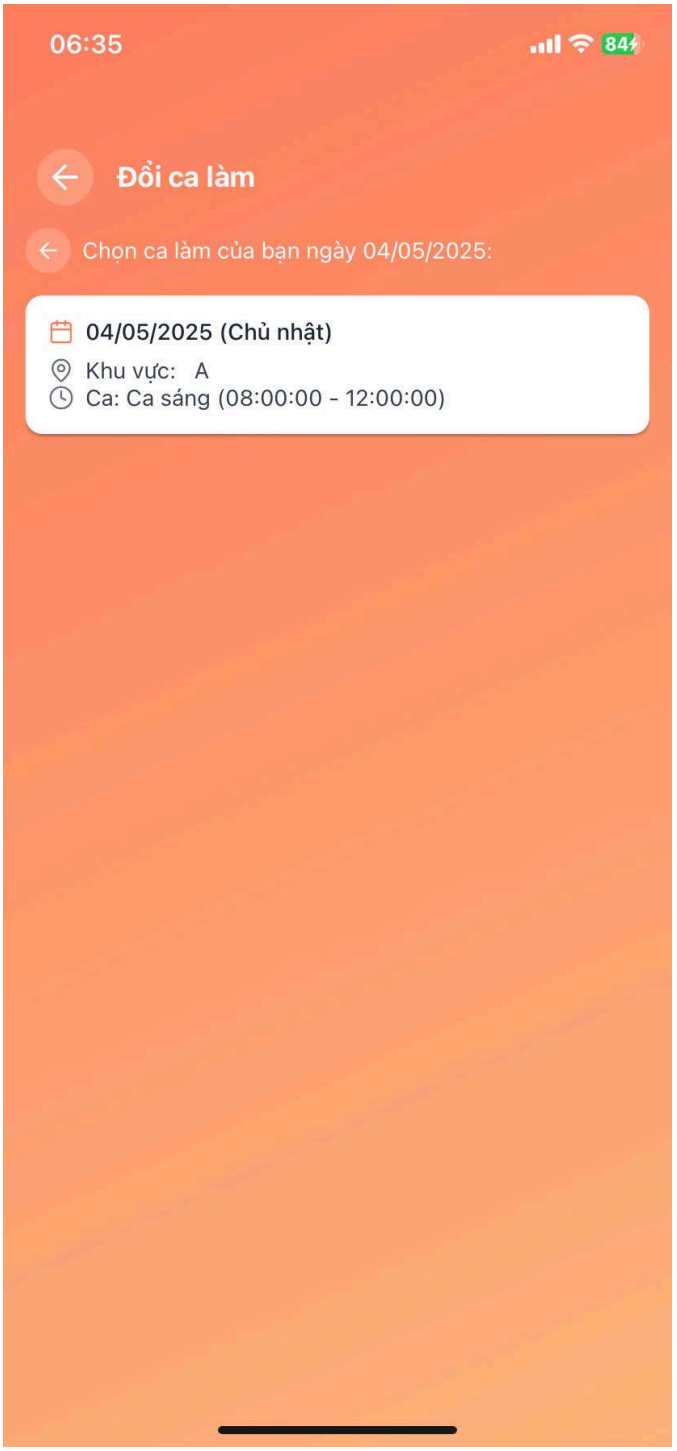
Item	Description
Purpose	Allow part-time employees to exchange approved shifts, with Manager oversight.
Actors	Employee A (Requester), Employee B (Receiver), Manager.

Prerequisites	Both employees have approved shifts.

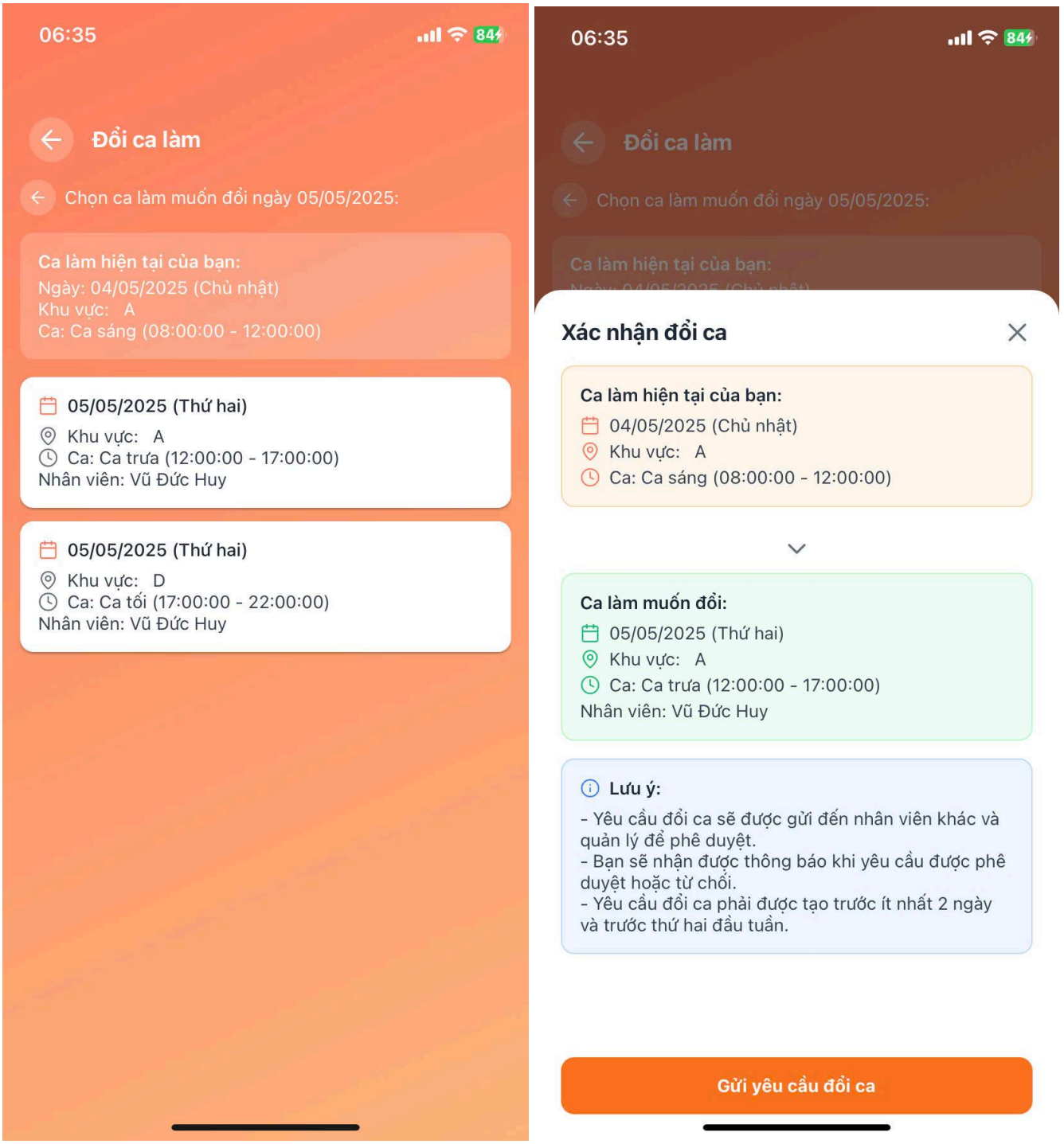
34. Employee A → **“Đổi ca làm”** in profile



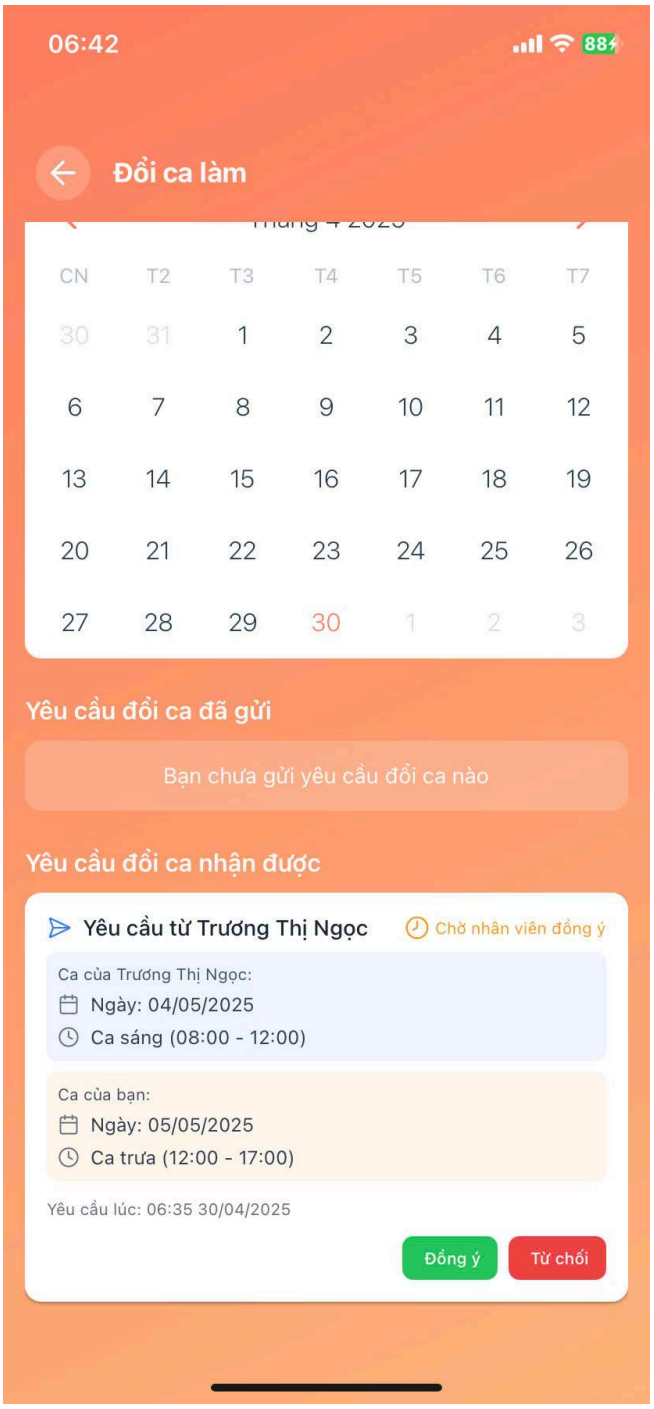
35. Selects date and WorkingSlot want to swap



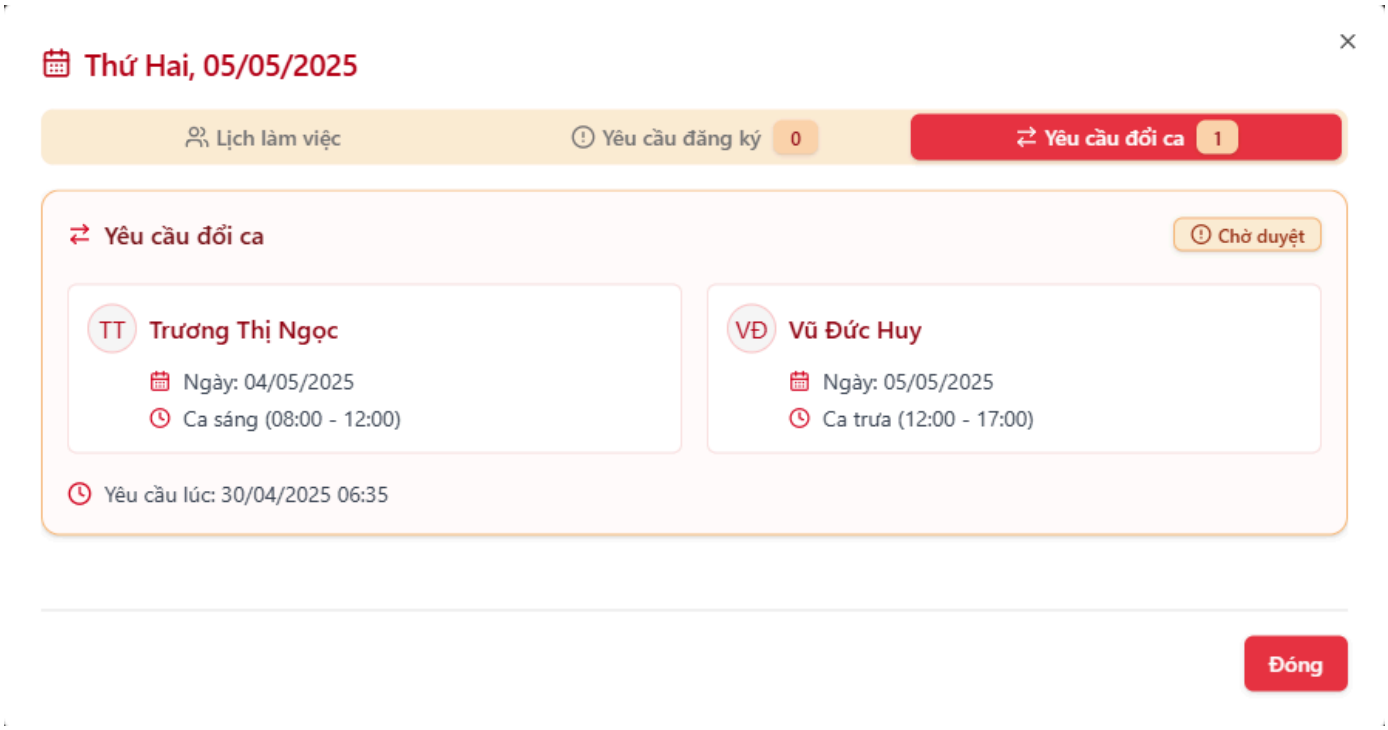
36. Selects shift to be swapped

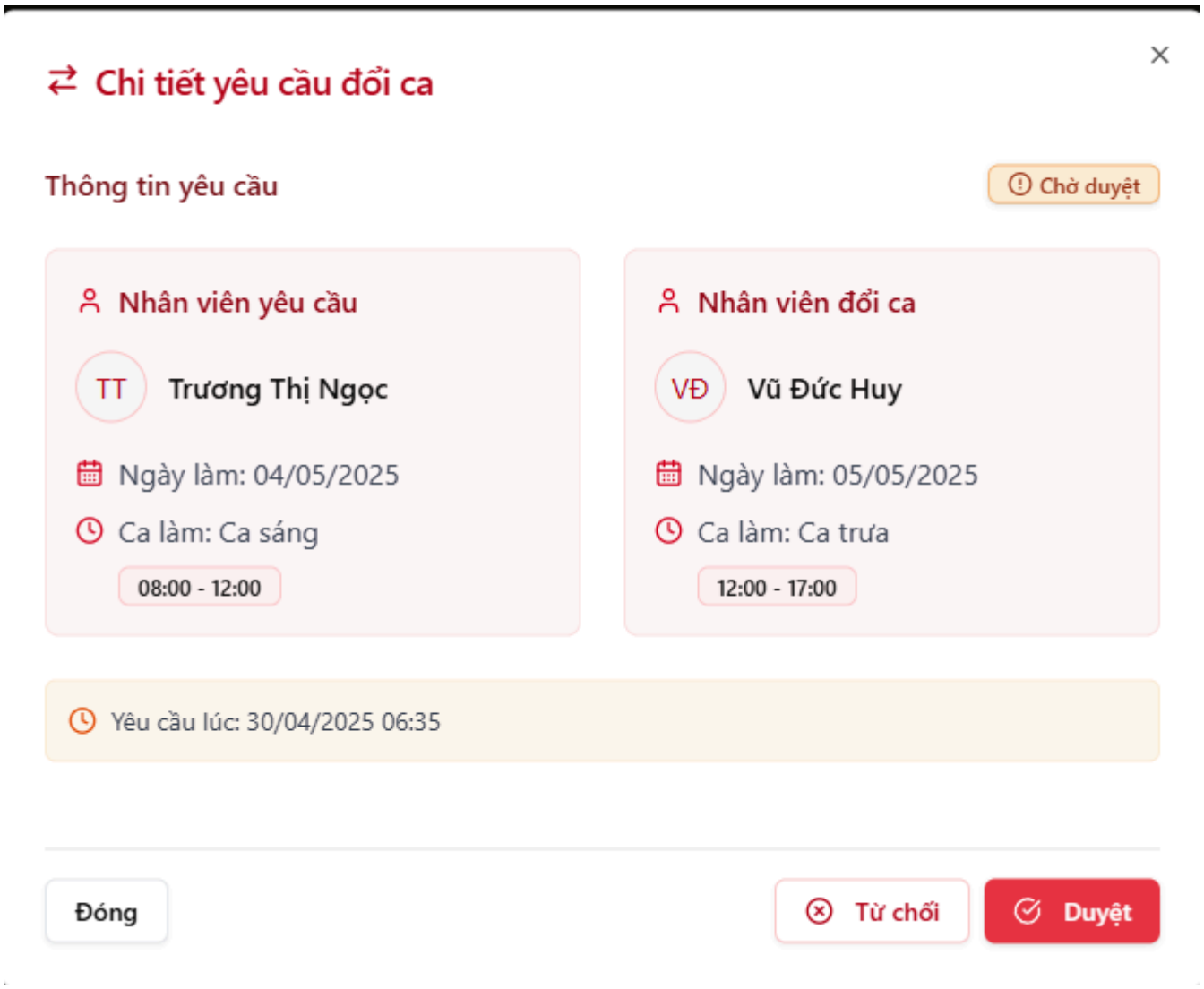


37. Employee B logs in and either clicks **“Đồng ý”** or **“Từ chối”**



37. Manager reviews and selects **“Duyệt”** or **“Từ chối”**


Thứ Hai, 05/05/2025
Lịch làm việc
Yêu cầu đăng ký 0
Yêu cầu đổi ca 1
Yêu cầu đổi ca
Chờ duyệt
TT
Trương Thị Ngọc
Ngày: 04/05/2025
Ca sáng (08:00 - 12:00)
VĐ
Vũ Đức Huy
Ngày: 05/05/2025
Ca trưa (12:00 - 17:00)
Yêu cầu lúc: 30/04/2025 06:35
Đóng


Chi tiết yêu cầu đổi ca
Thông tin yêu cầu
Chờ duyệt
Nhân viên yêu cầu
TT
Trương Thị Ngọc
Ngày làm: 04/05/2025
Ca làm: Ca sáng
08:00 - 12:00
Nhân viên đổi ca
VĐ
Vũ Đức Huy
Ngày làm: 05/05/2025
Ca làm: Ca trưa
12:00 - 17:00
Yêu cầu lúc: 30/04/2025 06:35
Đóng
Từ chối
Duyệt