MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Document

Smart Restaurant Reservation System – DineGO

FIVEGUYS		
Group Members	Hoang Huy Phuong	CE170428
	Tran Minh Thang	CE170148
	Duong Tuan Thanh	CE171822
	Nguyen Van Khoi	CE170049
	Huynh Doan Duc Sieu	CE170433
Supervisor	Ngo Giang Thanh	
Capstone Project code	SEP490_30	

- CanTho, August 2025 -

Table of Contents

Pages

Acknowledgement xiv

Definition and Acronyms xv

I. Project Introduction 1

1. Overview 1

2. Product Background 1

3. Existing Systems 2

4. Business Opportunity 6

5. Software Product Vision 6

6. Project Scope & Limitations 7

II. Project Management Plan 10

1. Overview 10

2. Management Approach 13

3. Project Deliverables 21

4. Responsibility Assignments 22

5. Project Communications 23

6. Configuration Management 24

III. Software Requirement Specification 33

1. Product Overview 33

2. Guest Requirements 38

3. Functional Requirements 207

4. Non-Functional Requirements 336

5. Requirement Appendix 337

IV. Software Design Description 340

1. Overall Description 340

2. System Architecture Design 341

3. Detailed Design 367

V. Software Testing Documentation 419

1. Scope of Testing 419

2. Test Strategy 419

3. Test Plan 420

4. Test Cases 421

5. Test Reports 422

VI. Release Package & User Guides 425
1. Deliverable Package 425
2. Installation Guides 425
3. User Manual 439
VII. Appendix 517

List of Figures

Pages

Figure 1: Home page of OpenTable website. 2
Figure 2: Home page of PasGo website. 3
Figure 3: Home page of Chope website. 4
Figure 4: Major features of the Smart Restaurant Reservation System (DineGo). 8
Figure 5: Software Development Lifecycle for DineGO 15
Figure 6: Team Roles and Collaboration Structure of the DineGO Project. 16
Figure 7: Software Development Lifecycle for DineGO 18
Figure 8: Selected Software Product Quality Characteristics 19
Figure 9: Document Repository Structure on Google Drive 24
Figure 10: Source Code Management Workflow of DineGO Project 25
Figure 11: ASP.NET logo representing the chosen backend framework 27
Figure 12: Microsoft SQL Server 2022 logo – database management system 28
Figure 13: Visual Studio Code logo – primary code editor for development 28
Figure 14: Draw.io logo – diagramming and modeling tool 28
Figure 15: Google Workspace icons – Docs, Sheets, Slides, and Forms 29
Figure 16: GitHub logo – version control and collaboration platform 29
Figure 17: VPS Hosting illustration – deployment infrastructure 29
Figure 18: Hugging Face logo – AI/ML model hosting and sharing platform 30
Figure 19: Python logo – programming language for AI model development 31
Figure 20: Amazon S3 logo – cloud storage service 31
Figure 21: Casso logo – automated payment processing platform integrated in DineGO 31
Figure 22: OpenStreetMap logo – open-source mapping 32
Figure 23: VietQR logo – QR payment solution in Vietnam 32
Figure 24: Context Diagram of the Smart Restaurant Reservation System – DineGO 33
Figure 25: Use Case diagram of Guest. 39
Figure 26: Use Case diagram of Customer. (1) 40
Figure 27: Use Case diagram of Customer. (2) 41
Figure 28: Use Case diagram of Restaurant Owner. (1) 42
Figure 29: Use Case diagram of Restaurant Owner. (2) 43
Figure 30: Use Case diagram of Admin. (1) 44
Figure 31: Use Case diagram of Admin. (2) 45
Figure 32: Screen Flow of Guest (Non-Login) 207
Figure 33: Screen Flow of Customer (Login) 208
Figure 34: Screen Flow of Restaurant Owner 209
Figure 35: Screen Flow of Admin 210
Figure 36: Entity Relationship Diagram of DineGO System 226
Figure 37: Customer login page of DineGO System 240
Figure 38: Guest register account page of DineGO System 241
Figure 39: Admin login account page of DineGO System 242
Figure 40: Forgot Password page of DineGO System 243
Figure 41: Browse Restaurants page of DineGO System 244
Figure 42: Restaurant Detail page of DineGO System 245
Figure 43: Food Viewing page of DineGO System 246

Figure 44: Food detail page of DineGO System 247
Figure 45: Map confirm of DineGO System 248
Figure 46: Cart page of DineGO System 249
Figure 47: Checkout page of DineGO System 250
Figure 48: View Order History page of DineGO System 251
Figure 49: View Order Detail page of DineGO System 252
Figure 50: Create Reservation page of DineGO System 254
Figure 51: View Reservation page of DineGO System 255
Figure 52: View Reservation payment of DineGO System 256
Figure 53: View/Edit Profile page of DineGO System 257
Figure 54: Change Password page of DineGO System 258
Figure 55: Apply as Owner page of DineGO System 259
Figure 56: Posts page of DineGO System 260
Figure 57: Blogs page of DineGO System 261
Figure 58: Post Detail page of DineGO System 262
Figure 59: Blog Detail page of DineGO System 263
Figure 60: Create Post page of DineGO System 264
Figure 61: Report page of DineGO System 265
Figure 62: Contact page of DineGO System 266
Figure 63: Notification of DineGO System 267
Figure 64: List Owner of DineGO System 268
Figure 65: Chat with owner of DineGO System 269
Figure 66: List Customer of DineGO System 270
Figure 67: Customer chat box of DineGO System 271
Figure 68: AI Assistant page of DineGO System 272
Figure 69: List of restaurant page of DineGO System 273
Figure 70: Chat with customer page of DineGO System 274
Figure 71: Restaurant Info page of DineGO System 275
Figure 72: Restaurant edit Info page of DineGO System 276
Figure 73: List of table page of DineGO System 277
Figure 74: Table manager page of DineGO System 278
Figure 75: Table Create manager page of DineGO System 279
Figure 76: Table edit manager page of DineGO System 280
Figure 77: Table delete manager page of DineGO System 281
Figure 78: Table order list page of DineGO System 282
Figure 79: Area list page of DineGO System 282
Figure 80: Area create page of DineGO System 283
Figure 81: Area edit page of DineGO System 284
Figure 82: Area delete page of DineGO System 285
Figure 83: Menu list page of DineGO System 286
Figure 84: Menu Create page of DineGO System 287
Figure 85: Menu edit page of DineGO System 288
Figure 86: Menu delete page of DineGO System 289
Figure 87: Food list page of DineGO System 290
Figure 88: Food create page of DineGO System 291
Figure 89: Food edit page of DineGO System 292
Figure 90: Food delete page of DineGO System 293

Figure 91: View Orders page of DineGO System 294
Figure 92: View Reservations page of DineGO System 295
Figure 93: Revenue by Reservation page of DineGO System 296
Figure 94: Dashboard page of DineGO System 297
Figure 95: Manage Restaurants page of DineGO System 298
Figure 96: Create restaurants page of DineGO System 299
Figure 97: Edit restaurants page of DineGO System 300
Figure 98: Block restaurants page of DineGO System 301
Figure 99: Delete restaurants page of DineGO System 302
Figure 100: License Approval page of DineGO System 302
Figure 101: Manage Customers page of DineGO System 303
Figure 102: Create Customers page of DineGO System 304
Figure 103: Edit Customers page of DineGO System 305
Figure 104: Block Customers page of DineGO System 306
Figure 105: Delete Customers page of DineGO System 307
Figure 106: Manage Vouchers page of DineGO System 308
Figure 107: Create Vouchers page of DineGO System 309
Figure 108: Edit vouchers page of DineGO System 310
Figure 109: Delete vouchers page of DineGO System 311
Figure 110: Category manager page of DineGO System 312
Figure 111: Create category page of DineGO System 313
Figure 112: Edit category page of DineGO System 314
Figure 113: Delete category page of DineGO System 315
Figure 114: Manage Notifications page of DineGO System 316
Figure 115: Create Notifications page of DineGO System 317
Figure 116: Edit notifications page of DineGO System 318
Figure 117: Delete notifications page of DineGO System 319
Figure 118: Restaurant owner page of DineGO System 319
Figure 119: Create restaurant owner page of DineGO System 320
Figure 120: Edit restaurant owner page of DineGO System 321
Figure 121: Delete restaurant owner page of DineGO System 322
Figure 122: Food order page of DineGO System 323
Figure 123: Table order page of DineGO System 324
Figure 124: Payment order page of DineGO System 325
Figure 125: Payment detail page of DineGO System 326
Figure 126: Blog manager page of DineGO System 327
Figure 127: Create blog page of DineGO System 328
Figure 128: Edit blog page of DineGO System 329
Figure 129: Delete blog page of DineGO System 330
Figure 130: Report manager page of DineGO System 331
Figure 131: Delete report page of DineGO System 332
Figure 132: Contact with customer manager page of DineGO System 333
Figure 133: Delete contact with customer page of DineGO System 334
Figure 134: System log page of DineGO System 335
Figure 135: Avatar update in admin page. 336
Figure 136: Overall Architecture Design 341
Figure 137: System Architecture 342

Figure 138: Package Diagram 343
Figure 139: sequence diagram cookie consent handling (Guest) 344
Figure 140: AWS S3 Image Upload. 345
Figure 141: AWS credentials and bucket detail are stored in appsettings.json 346
Figure 142: Database Design [1] 348
Figure 143: Class diagram authorize account 367
Figure 144: Sequence diagram login (Guest) 368
Figure 145: Sequence diagram login with google (Guest) 368
Figure 146: Sequence diagram register account (Guest) 368
Figure 147: Class diagram profile 369
Figure 148: Sequence diagram view profile (Customer) 369
Figure 149: Sequence diagram view profile (Admin) 370
Figure 150: Sequence diagram modify profile (Customer) 370
Figure 151: Class diagram of restaurant 371
Figure 152: Sequence diagram view list of restaurants (Guest) 371
Figure 153: Sequence diagram view restaurant list of restaurants (Customer) 372
Figure 154: Sequence diagram view restaurant list of restaurants (Admin) 372
Figure 155: Sequence diagram view restaurant detail (Customer) 372
Figure 156: Sequence diagram search restaurant (Guest) 372
Figure 157: Sequence diagram create restaurant (Restaurant Owner) 373
Figure 158: Sequence diagram create restaurant (Admin) 373
Figure 159: Sequence diagram modify restaurant (Restaurant Owner) 373
Figure 160: Sequence diagram modify restaurant (Admin) 374
Figure 161: Sequence diagram delete restaurant (Admin) 374
Figure 162: Class diagram of restaurant owner 375
Figure 163: Sequence diagram of register to be restaurant owner (Customer) 375
Figure 164: Class diagram of category 376
Figure 165: Sequence diagram view list of category (Admin) 376
Figure 166: Sequence diagram delete category(Admin) 377
Figure 167: Sequence diagram create category (Admin) 377
Figure 168: Sequence diagram modify category (Admin) 377
Figure 169: Class diagram of blog 378
Figure 170: Sequence diagram view list of blogs (User) 378
Figure 171: Sequence diagram view list of blogs (Admin) 378
Figure 172: Sequence diagram delete blog (Restaurant Owner) 379
Figure 173: Sequence diagram create blog (Admin) 379
Figure 174: Sequence diagram modify blog (Admin) 379
Figure 175: Class diagram of reservation 380
Figure 176: Sequence diagram view list of reservations (Customer) 381
Figure 177: Sequence diagram view list of reservations (Restaurant Owner) 381
Figure 178: Sequence diagram create reservation (Customer) 381
Figure 179: Sequence diagram modify reservation (Restaurant Owner) 382
Figure 180: Class diagram of post 383
Figure 181: Sequence diagram view list of posts (Customer) 383
Figure 182: Sequence diagram delete post (Customer) 384
Figure 183: Sequence diagram create post (Customer) 384
Figure 184: Sequence diagram modify post (Customer) 384

Figure 185: Class diagram of food 385
Figure 186: Sequence diagram search foods (Guest) 386
Figure 187: Sequence diagram search foods (Customer) 386
Figure 188: Sequence diagram Search foods by distance (Guest) 386
Figure 189: Sequence diagram search foods by distance (Customer) 387
Figure 190: Sequence diagram view list of foods (Guest) 387
Figure 191 Sequence diagram view list of foods (Customer) 387
Figure 192: sequence diagram view list of foods (Restaurant Owner) 388
Figure 193: Sequence diagram view food detail (Guest) 388
Figure 194: Sequence diagram view food detail (Customer) 388
Figure 195: sequence diagram create food (Restaurant Owner) 389
Figure 196: Sequence diagram modify food (Restaurant Owner) 389
Figure 197: Sequence diagram delete food (Restaurant Owner) 389
Figure 198: Class diagram of table 390
Figure 199: Sequence diagram view list of tables (Restaurant Owner) 390
Figure 200: Sequence diagram create table (Restaurant Owner) 391
Figure 201: Sequence diagram modify table (Restaurant Owner) 391
Figure 202: Sequence diagram delete table (Restaurant Owner) 391
Figure 203: Sequence diagram AI assistant module - Suggest Tag (Guest) 392
Figure 204: Sequence diagram AI assistant module - Response Generation (User) 392
Figure 205: Class diagram AI assistant module 393
Figure 206: data sample 395
Figure 207: result 397
Figure 208: Sample training data for AI response generation 398
Figure 209: Example interaction and AI-generated food suggestion 400
Figure 210: Class diagram for Order module 401
Figure 211: Sequence diagram view list of orders food (Customer) 402
Figure 212: Sequence diagram view list of orders food (Restaurant Owner) 402
Figure 213: Sequence diagram view history order detail (Customer) 402
Figure 214: Sequence diagram create order(Customer) 403
Figure 215: Sequence diagram modify order(Restaurant Owner) 403
Figure 216: Class diagram voucher 404
Figure 217: Sequence diagram view list of vouchers (Admin) 404
Figure 218: Sequence diagram create voucher (Admin) 405
Figure 219: Sequence diagram modify voucher (Admin) 405
Figure 220: Sequence diagram delete voucher (Admin) 405
Figure 221: Class diagram verification 406
Figure 222: Sequence diagram view list of verifications (Admin) 406
Figure 223: Sequence diagram approve verification (Admin) 407
Figure 224: Sequence diagram reject verification (Admin) 407
Figure 225: Class diagram notification 408
Figure 226: Sequence diagram view list notifications (Admin) 408
Figure 227: Sequence diagram create notification (Admin) 409
Figure 228: Sequence diagram modify notification (Admin) 409
Figure 229: Sequence diagram delete notification (Admin) 409
Figure 230: Class diagram cart 410
Figure 231: Sequence diagram view cart (Customer) 410

Figure 232: Sequence diagram add food to cart (Customer) 411
Figure 233: Sequence diagram modify cart item quantity (Customer) 411
Figure 234: Sequence diagram Delete food in cart (Customer) 411
Figure 235: Class diagram of menu 412
Figure 236: Sequence diagram view list of menus by restaurant (Guest) 412
Figure 237: Sequence diagram view list of menus by restaurant (Restaurant Owner) 412
Figure 238: Sequence diagram create menu (Restaurant Owner) 413
Figure 239: Sequence diagram modify menu (Restaurant Owner) 413
Figure 240: Sequence diagram delete menu (Restaurant Owner) 413
Figure 241: Class diagram of menu 414
Figure 242: Sequence diagram view list of customers (Admin) 414
Figure 243: Sequence diagram create customer (Admin) 414
Figure 244: Sequence diagram delete customer (Admin) 415
Figure 245: Sequence diagram modify customer (Admin) 415
Figure 246: Sequence diagram block customer (Admin) 415
Figure 247: Class diagram of rating 416
Figure 248: Sequence diagram add or update rating (Customer) 416
Figure 249: Class diagram of rating 417
Figure 250: Sequence diagram view list of logs (Admin) 417
Figure 251: Class diagram of chat 418
Figure 252: Sequence diagram Chat between Customers (Customer) 418
Figure 253: Unit Test case UC List 423
Figure 254: Unit Test Case percentage chart 424
Figure 255: System test case report 424
Figure 256: Install SQL Server 2022 and SSMS 2019 426
Figure 257: Accepting the license terms 426
Figure 258: Specifying installation location and starting installation 427
Figure 259: installation completed with option to install SSMS 427
Figure 260: Selecting location and starting installation 428
Figure 261: Installation completed successfully 428
Figure 262: Backup database 429
Figure 263: Selecting device and adding backup 429
Figure 264: Selecting the backup file (.bak) 430
Figure 265: Confirming the selected backup 430
Figure 266: Final confirmation 431
Figure 267: Install .NET Hosting Bundle 5.0.17 431
Figure 268: Wait install process 432
Figure 269: Install complete 432
Figure 270: Install IIS Manager 433
Figure 271: Selecting the destination server 433
Figure 272: Server role selection in Windows Server 2022 434
Figure 273: PowerShell command for installing Web Server (IIS) 434
Figure 274: Detailed selection of Web Server (IIS) role 435
Figure 275: Adding a new website in Internet Information Services (IIS) Manager 435
Figure 276: Selecting the "SEP490_API" folder 436
Figure 277: Configuring the "Add Website" dialog in IIS 437
Figure 278: Selecting the "SEP490_FE" folder 437

Figure 279: Configuring the "Add Website" dialog in IIS..... 438
Figure 280: Web page 438
Figure 281: Register page of DineGO 441
Figure 282: Login page of DineGO..... 442
Figure 283: Login with Google..... 442
Figure 284: Login admin page of DineGO..... 443
Figure 285: Profile page of DineGO..... 444
Figure 286: Home page of DineGO..... 444
Figure 287: Change password page 445
Figure 288: Logout function of DineGO 445
Figure 289: Admin logout function of DineGO..... 446
Figure 290: Forgot password function 446
Figure 291: Forgot password function 447
Figure 292: Gmail reset password..... 448
Figure 293: View customer list of DineGO 448
Figure 294: Customer list of DineGO..... 449
Figure 295: Add customer form of DineGO..... 449
Figure 296: Customer list of DineGO..... 450
Figure 297: Edit customer form of DineGO..... 450
Figure 298: View customer list of DineGO 451
Figure 299: Block form of DineGO..... 451
Figure 300: View customer list of DineGO 452
Figure 301: Unblock form of DineGO 452
Figure 302: View restaurant owner list of DineGO 453
Figure 303: View restaurant owner page of DineGO 454
Figure 304:. Add restaurant owner page of DineGO..... 454
Figure 305: View restaurant owner page of DineGO 454
Figure 306: Update restaurant owner page of DineGO 455
Figure 307: View restaurant owner page of DineGO 455
Figure 308: View restaurant owner page of DineGO 456
Figure 309: Business license page of DineGO 456
Figure 310: Business license page of DineGO 457
Figure 311: Business license page 458
Figure 312: Reject business license page 458
Figure 313: View restaurant page of DineGO 459
Figure 314: View restaurant page of DineGO 459
Figure 315: Create restaurant of DineGO 460
Figure 316: View restaurant page of DineGO 460
Figure 317: Block restaurant page of DineGO..... 461
Figure 318: View restaurant page of DineGO 461
Figure 319: Update restaurant form of DineGO 462
Figure 320: View restaurant page of DineGO 463
Figure 321: Confirm delete restaurant page..... 463
Figure 322: View category list of DineGO 464
Figure 323: Category page of DineGO..... 464
Figure 324: Create category page of DineGO..... 465
Figure 325: Category page in DineGO 465

Figure 326: Update category in DineGO 466
Figure 327: Category page in DineGO 466
Figure 328: Delete category process in DineGO 467
Figure 329: View order list of DineGO 467
Figure 330: View order list of DineGO 468
Figure 331: Table reservation in DineGO 469
Figure 332: Payment manage page in DineGO 469
Figure 333: Payment manages page in DineGO 470
Figure 334: Print invoice in DineGO 470
Figure 335: Blog manages page in DineGO 471
Figure 336: Blog manages page in DineGO 471
Figure 337: Add blog form in DineGO 472
Figure 338: Blog manages page in DineGO 472
Figure 339: Edit blog form in DineGO 473
Figure 340: Blog manage page in DineGO 473
Figure 341: Delete confirm blog in DineGO 474
Figure 342: Voucher page in DineGO 475
Figure 343: Voucher page in DineGO 475
Figure 344: Create voucher in DineGO 475
Figure 345: Voucher page in DineGO 476
Figure 346: Update voucher in DineGO 476
Figure 347: Voucher page in DineGO 477
Figure 348: Delete voucher in DineGO 477
Figure 349: Notification page in DineGO 478
Figure 350: Notification page in DineGO 478
Figure 351: Add notification form of DineGO 479
Figure 352: Notification page in DineGO 479
Figure 353: Update notification form of DineGO 480
Figure 354: Notification page in DineGO 481
Figure 355: Delete notification in DineGO 481
Figure 356: Dashboard and report in DineGO 482
Figure 357: View activity log in DineGO 482
Figure 358: Search restaurant feature 483
Figure 359: Select location feature 483
Figure 360: Restaurant page of DineGO 484
Figure 361: Restaurant detail page of DineGO 484
Figure 362: Menu of restaurant page of DineGO 485
Figure 363: Food detail page of DineGO 485
Figure 364: Add to cart button on restaurant menu page 486
Figure 365: Shopping cart page of DineGO 486
Figure 366: Track order delivery page of DineGO 487
Figure 367: Make table reservation feature of DineGO 488
Figure 368: Restaurant detail page 488
Figure 369: Make table reservation form of DineGO 489
Figure 370: Payment for make table reservation of DineGO 489
Figure 371: Profile page of DineGO 490
Figure 372: Reservation history of DineGO 490

Figure 373: Homepage of DineGO 491
Figure 374: Post function of DineGO 491
Figure 375: Form create post of DineGO 492
Figure 376: Social media page of DineGO 492
Figure 377: Feed in social media of DineGO 493
Figure 378: Social media page of DineGO 493
Figure 379: Social media page of DineGO 494
Figure 380: Friend list in chat panel 495
Figure 381: Sending message to a friend 495
Figure 382: Add Friend screen 496
Figure 383: Friend list showing pending request 496
Figure 384: Chat window 497
Figure 385: Profile page of DineGO 497
Figure 386: Register restaurant owner form of DineGO 498
Figure 387: View Vouchers and Points 498
Figure 388 : Voucher page 499
Figure 389: Reedeem point for vouchers 499
Figure 390: Voucher page 500
Figure 391 : Point transaction history page 500
Figure 392: Create restaurant profile feature 501
Figure 393: Create restaurant profile form 501
Figure 394: Update restaurant feature 502
Figure 395: Update restaurant profile feature 502
Figure 396: Update restaurant profile feature 503
Figure 397: Menu page of DineGO 504
Figure 398: Menu page of DineGO 504
Figure 399: Menu page of DineGO 505
Figure 400: Menu page of DineGO 505
Figure 401: Menu page of DineGO 506
Figure 402: Menu page of DineGO 506
Figure 403: Confirm delete menu 507
Figure 404: Menu page of DineGO 507
Figure 405: View food in menu 508
Figure 406: Food page in menu 508
Figure 407: Add food feature 509
Figure 408: View food in menu 509
Figure 409: Update food feature 510
Figure 410: View food in menu 511
Figure 411: Delete food feature 511
Figure 412: Handle reservation feature 511
Figure 413: Detail table 512
Figure 414: Handle reservation feature 512
Figure 415: Process food order feature 513
Figure 416: Respond to customer inquiries feature in DineGO 513
Figure 417: View statics dashboard in DineGO 514
Figure 418: View business license in DineGO 515
Figure 419: Submit business license feature in DineGO 516

Figure 420: Submit business license feature in DineGO 516

List of Tables

Pages

Table 1: Benefits and drawbacks of existing systems. 5
Table 2: Project Work Breakdown Structure (WBS) and Effort Estimation 10
Table 3: Software Quality Targets per Testing Plan. 12
Table 4: Project Risk Assessment and Response Plan 13
Table 5: DineGO Agile Sprint Schedule and Key Development Activities 14
Table 6: Additional Support Roles (optional, depending on team structure) 20
Table 7: Training & Development Plan 20
Table 8: Project Deliverables Schedule 21
Table 9: Responsibility Assignment Matrix (RAM) 22
Table 10: Project Communication Plan 23
Table 11: Tools and Infrastructure Overview 27
Table 12: Business Rules for the DineGO System 34
Table 13: Description of System Actors 38
Table 14: Use Case Descriptions for system 46
Table 15: Screen Descriptions for Guest and Authentication Features 211
Table 16: Non-Screen System Functions of the DineGO Platform 225
Table 17: List of Application Messages 338
Table 18 :Package Descriptions 343
Table 19: Training strategy table 395
Table 20: Test level table 420
Table 21: Supporting tools table 420
Table 22: Human resources table 421
Table 23: Test environment table 421
Table 24: Test milestones table 421
Table 25: Deliverable Package 425
Table 26: Terms and definitions 439

Acknowledgement

We would like to express our deepest gratitude to Mr. Ngo Giang Thanh, our supervisor, for his valuable guidance, encouragement, and continuous support throughout the development of the DineGO project. His insightful feedback and expertise have been instrumental in helping our team shape the system into its current form.

We also extend our sincere thanks to the faculty and staff of the university for providing us with the necessary resources, facilities, and a conducive learning environment that allowed us to carry out this project effectively.

Special appreciation is given to all the lecturers who have taught and guided us through nine semesters of study. The knowledge, skills, and experiences gained during these nine semesters have laid a solid foundation for us to apply in the development of this project.

Lastly, we are grateful to all our teammates for their dedication, hard work, and collaboration, which have been key factors in completing this project successfully.

Definition and Acronyms

Acronym	Definition
DineGO	Smart Restaurant Reservation System – DineGO
AWS	Amazon Web Services
BA	Business Analysis
BR	Business Rule
ERD	Entity Relationship Diagram
SDD	Software Design Description
SPMP	Software Project Management Plan
SRS	Software Requirement Specification
UAT	User Acceptance Test
UC	Use Case
API	Application Program Interface

I. Project Introduction

1. Overview

1.1 Project Information

- Project name: Smart Restaurant Reservation System – DineGO
- Project code: SEP490_30
- Group name: FIVEGUYS
- Software type: Website

1.2 Project Team

Full Name	Role	Email	Mobile
Ngo Giang Thanh	Lecturer	thanhng4@fpt.edu.vn	0946344233
Hoang Huy Phuong	Leader	phuonghhce170428@fpt.edu.vn	0326776754
Tran Minh Thang	Member	thangtmce170148@fpt.edu.vn	0362336362
Nguyen Van Khoi	Member	khoinvce170049@fpt.edu.vn	0384077921
Huynh Doan Duc Sieu	Member	sieuhddce170433@fpt.edu.vn	0366230497
Duong Tuan Thanh	Member	thanhdtce171822@fpt.edu.vn	0941852832

2. Product Background

In the context of rapidly advancing digital technologies and growing demand for personalized customer service, smart restaurant reservation systems are becoming increasingly essential. Traditionally, customers had to make phone calls or visit restaurants in person to reserve a table, which often led to inconvenience such as long waiting times, miscommunication, overbooking, or no availability during peak hours. Likewise, restaurants faced difficulties managing reservations efficiently, resulting in poor service coordination and staffing issues.

To address these problems, the proposed platform introduces an intelligent online reservation system that allows users to book tables and pre-order dishes directly through the website. Customers can select available time slots, table locations, number of guests, and preferred menu items—all in real time—streamlining their dining experience and reducing uncertainty.

Beyond basic booking functionality, the platform aims to become a community-driven dining hub by enabling users to share their experiences through interactive posts. Diners can post reviews, photos, and comments about the restaurant, food, and service, helping others discover and evaluate dining options through real user feedback.

A key innovation of the platform is the integration of Artificial Intelligence (AI) to enhance personalization and user engagement. The AI module is designed to recommend dishes based on users' search queries and their interactions with menu items. When a user searches for food using the search bar, the AI analyzes the entered keywords to extract relevant tags using natural language processing. These tags are then recorded and their priority is increased for that user. Additionally, every time a user clicks to view the detail of a dish, the system increases the priority of the tags associated with that dish for the user. Over time, the system learns from each user's search and

interaction history, continuously updating their preferences. As a result, the food recommendations become increasingly personalized, highlighting dishes and restaurants that best match the user's tastes and interests. This dynamic approach ensures that the more a user searches for or interacts with certain types of dishes, the more likely those dishes (or similar ones) will be recommended in the future.

This solution was envisioned in response to feedback from both end-users seeking more convenient, engaging dining services and restaurant operators looking for smarter tools to manage their business and enhance customer satisfaction.

3. Existing Systems

3.1 OpenTable

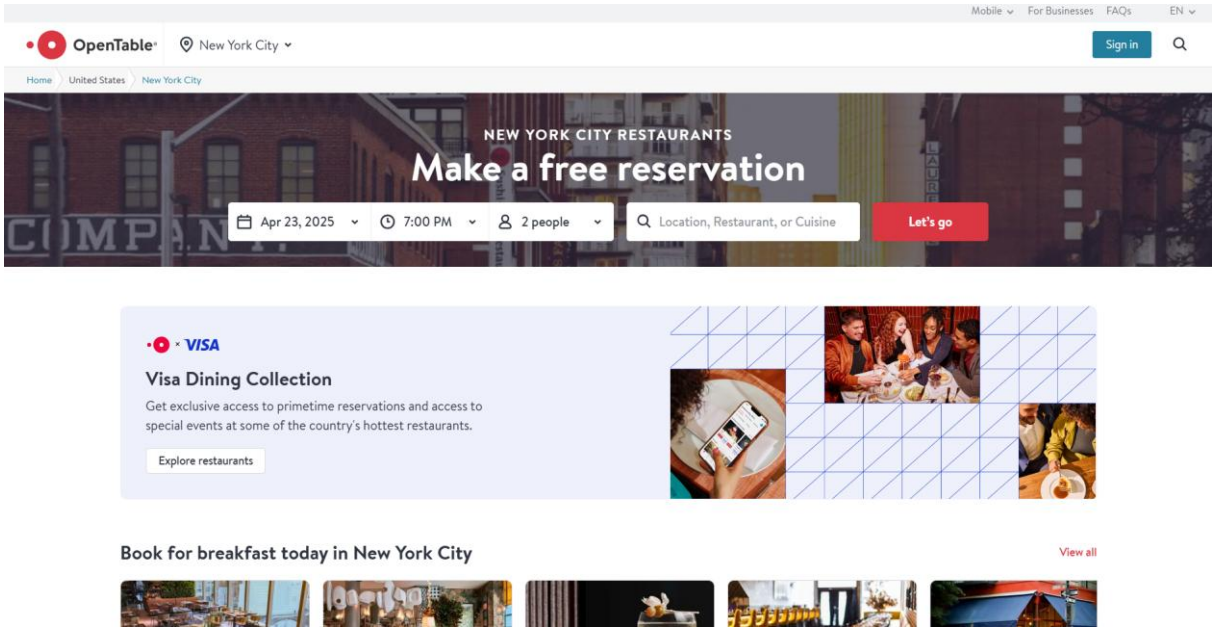


Figure 1: Home page of OpenTable website.

In 1999, the website or portal was launched and initially offered only a limited selection of restaurants in San Francisco. OpenTable has more than 37,000 restaurant customers and has placed more than 760 million diners around the world since its inception. The OpenTable service is available in the United States, Canada, Germany, Japan, Mexico, and the United Kingdom. OpenTable also operates TopTable.com, a leading restaurant reservation site in the UK.

OpenTable's reservation service helps restaurants accept the reservations and thus fill more tables. Additionally, restaurants can use OpenTable's electronic reservation book to completely replace the paper reservation book. The system modernizes reservation and guest management and allows restaurants to build up a detailed guest database, i.e. to achieve higher guest recognition and to conduct targeted email marketing. For restaurants that tend to work less with reservations, OpenTable offers the OT-Connect product. A web-based service that still offers restaurants the option to accept online reservations.

For restaurant-goers, hotel concierges and administrative staff, the website offers a method to find tables available for a specific time of day that meet the desired criteria such as cuisine, price range and location. Guests can also make reservations through apps for most smartphone operating systems.[3]

a. **Benefits:** (Based on personal assessment)

Modern interface, easy to use for both users and restaurants.

Large database, allowing to search and book tables in many countries, many types of restaurants.

Integrated reviews, comments, table booking history - increase transparency and reliability.

Provide CRM (customer management) tools for restaurants to track user behavior and optimize services.

b. **Drawbacks:** (Based on personal assessment)

The service fee is quite high for restaurants when joining the platform.

There is no good support feature for small and medium-sized restaurants or those without a technology platform.

3.2 PasGo

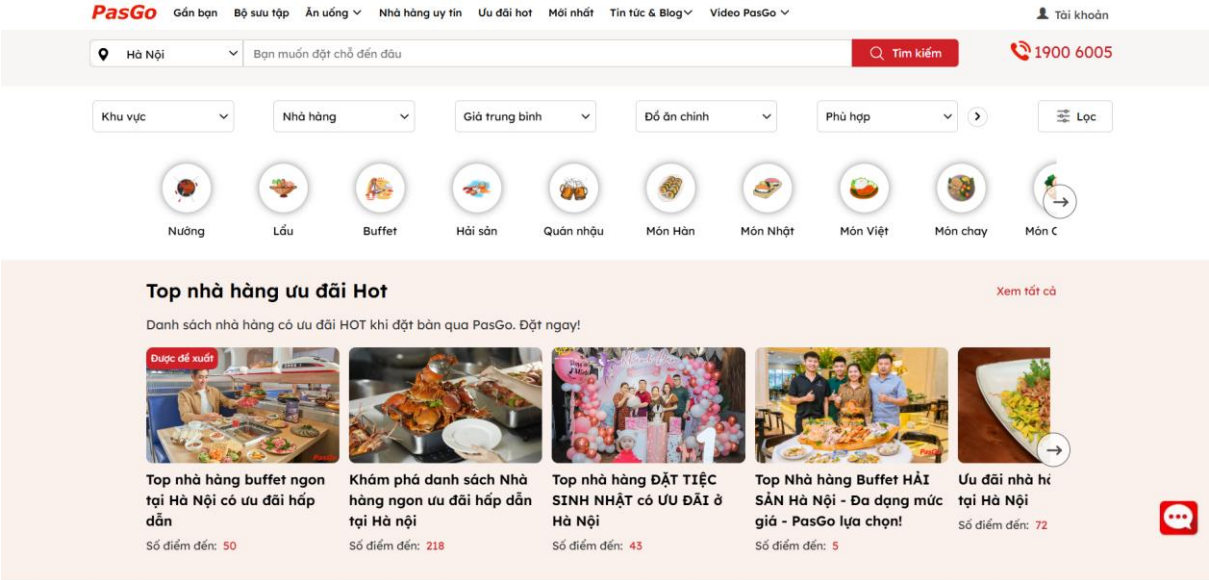


Figure 2: Home page of PasGo website.

PasGo is a prominent online restaurant table reservation platform in Vietnam, especially popular in big cities such as Hanoi, Ho Chi Minh City, Da Nang, and Hai Phong that helps customers easily search and choose restaurants according to their preferences and nearest locations. With optimized features and a friendly interface, PasGo not only simplifies the table setup process but also opens up a variety of culinary experiences. The platform is being improved based on more than 10 years of experience in developing and applying automation technologies, ongoing quantity transformation to bring special value to both users and restaurant partners.

a. **Benefits:** (Based on personal assessment)

User-friendly interface, easy to use, suitable for both young and old people. Convenient filters by area, dish type, price, promotion, review, help find restaurants quickly.s

Integrate many useful features: online table reservation, delivery order, accumulate member points, give promotional codes.

Large database, diverse types, large restaurant partners

Good customer support via hotline or direct messaging with customer care department.

b. **Drawbacks:** (Based on personal assessment)

User experience is not smooth, the application may lag or freeze, especially during peak hours.

User reviews and comments are not censored, can be misleading.

Not popular in many provinces, mainly serving large cities, less options in provinces and towns.

Privacy policy is not transparent, there is no transparency in data protection policy, users.

3.3 Chope

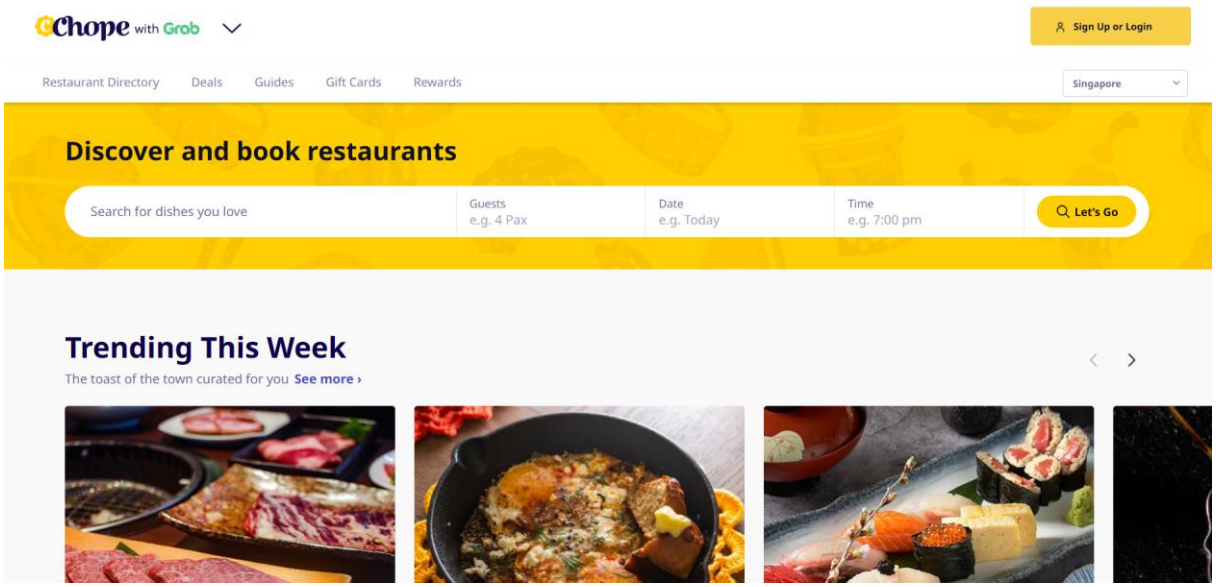


Figure 3: Home page of Chope website.

Founded in Singapore in 2011, Chope is an online restaurant booking platform that aims to simplify dining out across Asia. Operating in major Asian cities such as Singapore, Bangkok, Hong Kong, Jakarta, Bali, and Shanghai, Chope offers both diners and restaurants a highly efficient and digital-first approach to dining reservations.

Chope enables users to search restaurants, book tables in real-time, and access exclusive promotions through the website or mobile apps. The platform's selling point lies in its integration of ChopeDeals, offering discounts and special set menus for users who book through the platform.

On the restaurant side, ChopeBook provides a full-featured table management and CRM solution that helps restaurants handle reservations, track customer behavior, and optimize service.

a. **Benefits:** (Based on personal assessment)

Clean, modern interface

Deals and promos enhance user appeal

Regional coverage across Asia

Helps restaurants with CRM and guest management

b. **Drawbacks:** (Based on personal assessment)

Focused mostly on big cities

Some restaurants report high commission fees

Occasionally inaccurate table availability

Table 1: Benefits and drawbacks of existing systems.

Feature	OpenTable	PasGo	Chope
Online reservation website	Yes	Yes	Yes
Mobile app support	iOS & Android	iOS & Android	iOS & Android
Location-based search/filtering	Yes	Yes	Yes
Filter by cuisine/price/type	Yes	Yes	Yes
Promotions / Discount deals	Limited	Strong local deals	Strong (ChopeDeals)
Real-time table availability	Yes	Yes	Yes
Auto-confirmation & reminders	Yes (via email/app)	Available but basic	Yes
User reviews & ratings	Integrated and verified	Present but not moderated	Available
CRM tools for restaurants	Strong (OpenTable CRM)	Minimal or not available	Available (ChopeBook)
Table management system	Yes	Not clearly disclosed	Yes (ChopeBook)
Customer support service	Web/email based only	Hotline and live chat	Limited direct support

Feature	OpenTable	PasGo	Chope
Loyalty points / membership system	Not highlighted	Available	Available (Chope-Dollars)
Geographical availability	International (US, UK, Japan, etc.)	Vietnam (main cities only)	Asia (Singapore, Bangkok, HK, etc.)

4. Business Opportunity

The demand for advance table reservations at restaurants and eateries is increasing significantly, especially in urban areas where people frequently dine out during weekends, holidays, or peak hours. However, most restaurants still rely on traditional booking methods such as phone calls or walk-ins, which can be inconvenient for customers. This often leads to long waiting times, uncertainty about table availability, or the inability to make reservations outside of business hours.

An online table reservation system can solve these issues by allowing customers to check **real-time table availability**, make bookings for a desired time, and receive instant confirmation. The system will also **integrate with map services such as Google Maps** to help users easily find the nearest available restaurants based on their current location or a selected area. This feature is especially useful in dense urban areas where time and convenience are critical factors in customer decision-making.

To enhance interactivity and support, the system will include a built-in **AI chatbot** that assists users to answer frequently asked questions. This AI assistant not only improves customer experience but also reduces pressure on manual support staff.

At the same time, restaurants benefit from optimized space usage, reduced no-shows or late arrivals, and better demand forecasting for staff and inventory planning.

While some table booking platforms exist in the market, many do not adequately address the needs of local markets or lack customization options for small and medium-sized restaurants. The proposed system offers a flexible, easily integrated solution tailored to the Vietnamese market, supporting both end-users (customers) and restaurant managers. Additional features such as real-time notifications, post-visit feedback, personalized promotions, location-based search, AI-powered assistance, and user preference analytics will further enhance its competitiveness and align with the ongoing digital transformation trends in the food service industry.

5. Software Product Vision

For customers who want a convenient and efficient way to reserve tables at restaurants, the Restaurant Table Reservation System is a web and mobile application that allows users to search for restaurants, view real-time table availability, make bookings, and receive instant confirmation.

Unlike traditional booking methods via phone or walk-in, this system saves time, reduces the risk of not finding a seat, and enhances the overall customer experience. For restaurants, it helps manage reservations efficiently, minimizes errors in table arrangements, and provides analytics to improve service quality.

The system is designed to support the digitalization goals of the food service industry and the growing demand for personalized customer experiences. In the future, it can be expanded to include additional features such as pre-ordering food, online payments, and integration with delivery platforms to serve both dine-in and remote customers.

6. Project Scope & Limitations

6.1 Major Features

FE-01: User Management

Supports customer account management, including registration, login/logout, and personal profile updates.

FE-02: Restaurant Management

Allows restaurant owners to register their business, create and update restaurant profiles, and maintain essential details such as contact information and licenses.

FE-03: Menu Management

Enables restaurant owners to add, edit, or remove food items. Each dish can include a description, price, and image.

FE-04: Table & Reservation Management

Provides customers with real-time table booking and cancellation. Restaurant owners can manage seating layouts and reservation schedules efficiently.

FE-05: Order Management

Handles online food orders, whether dine-in, take-away, or pre-order. Customers can add special requests, and restaurants can process orders instantly.

FE-06: Payment & Transaction Management

Supports multiple payment methods (cards, QR, e-wallets), tracks transaction history, and allows resolution of payment issues.

FE-07: Loyalty & Gift Card Management

Manages customer loyalty points, redemption for rewards, and digital or physical gift cards.

FE-08: Social & Community Management

Includes short video sharing, user interactions (likes, comments, follows), and messaging between customers and restaurants.

FE-09: Promotion & Advertisement Management

Allows restaurants to create promotional campaigns, manage discounts, and track advertisement performance.

FE-10: Admin Management

Allows administrators to manage users, review and remove inappropriate content, verify restaurants, and handle system-level operations.

Figure 4: Major features of the Smart Restaurant Reservation System (DineGo).

6.2 Limitations & Exclusions

LI-01:Mobile App Platform

The DineGo system does not currently support mobile app platforms (iOS/Android).

LI-02: Offline Reservation Methods

Offline reservation methods (e.g., phone call, SMS, in-person booking) are outside the scope of the system. All reservations must be performed online through the DineGo platform.

LI-03: AI Chatbot Scope

The AI chatbot is limited to providing suggestions and basic support; it does not make decisions or place orders/reservations on behalf of users.

LI-04: Payment Processing
In-app payment processing supports only selected domestic methods (e.g., MoMo, bank cards, QR codes). International payment gateways (e.g., PayPal, Stripe) are not supported at this time.

LI-05: Deployment Scope
The system is designed for use by participating restaurants registered within the local region (e.g., Vietnam). It is not intended for global-scale deployment in this version.

LI-06: Menu Filtering
Custom dietary or allergy-specific filtering in the menu search function is not yet supported. Menu tagging and filtering are limited to general categories (e.g., vegetarian, spicy, price range).

LI-07: Multi-language Support
The current system only supports a single default language (e.g., Vietnamese/English). Multi-language translation is not yet available.

LI-08: Third-Party Delivery Services
Integration with external delivery services (e.g., GrabFood, ShopeeFood) is not supported. Only in-house delivery tracking is available.

LI-09: Promotions & Marketing Content
Promotion campaigns are limited to form mechanisms such as pop-ups or banners. The system does not yet support in-depth marketing content management (e.g., blogs, email marketing, social media integration, or ad personalization).

LI-10: Social Content Moderation
Content moderation for social posts and comments is fully dependent on admin intervention. No role-based moderation (e.g., restaurant owners, moderators) or automated AI filtering is currently supported.

LI-11: Order Payment Limitation
Currently, order payments do not support online banking or digital wallets. Customers can only pay via Cash on Delivery (COD).

II. Project Management Plan

1. Overview

1.1 Scope & Estimation

The DineGO project aims to develop a smart online restaurant table reservation system, including customer, restaurant owner, and admin functions. The table below estimates the main tasks and assigns responsibilities to each member:

Table 2: Project Work Breakdown Structure (WBS) and Effort Estimation

#	WBS Item	Complexity	Est. Effort (man-day)
1	**Project Planning & Documentation**		**55**
1.1	WBS & Schedule Management	Simple	1
1.2	Create Reports (1–7)	Complex	24
1.2.1	Create Report 1	Simple	2
1.2.2	Create Report 2	Simple	3
1.2.3	Create Report 3	Medium	5
1.2.4	Create Report 4	Medium	5
1.2.5	Create Report 5	Medium	3
1.2.6	Create Report 6	Medium	3
1.2.7	Create Report 7	Medium	3
1.3	Review & Finalize Documentation	Medium	6
2	**Database & System Design**		**36**
2.1	ERD Database	Simple	4
2.2	Database Table Design	Medium	6
2.2.1	Define Table Schema	Medium	2
2.2.2	Define Relationships & FKs	Medium	2
2.2.3	Normalize Database	Medium	2
2.3	System Architecture & Diagram	Complex	10
2.3.1	Design System Components	Complex	4
2.3.2	Draw Component & Deployment Diag	Complex	3
2.3.3	Review & Refine Architecture	Complex	3
3	**Implementation & Integration**		**70**

#	WBS Item	Complexity	Est. Effort (man-day)
3.1	Coding Main Modules	Complex	15
3.1.1	Build Core Business Logic	Complex	6
3.1.2	Implement UI & Controllers	Complex	5
3.1.3	Develop API/Service Layer	Complex	4
3.2	Integration of Modules	Medium	10
3.2.1	Develop Frontend & Backend	Medium	4
3.2.2	Integrate Internal Services	Medium	3
3.2.3	Integrate External Services	Medium	3
3.3	Debugging & Refactoring	Medium	10
3.3.1	Bug Fixing After Testing	Medium	5
3.3.2	Code Refactoring & Optimization	Medium	5
4	**Testing & Finalization**		**32**
4.1	Create Test Cases	Medium	6
4.1.1	Unit Test Cases	Medium	2
4.1.2	Integration Test Cases	Medium	2
4.1.3	Acceptance Test Scenarios	Medium	2
4.2	Testing (Unit, Integration, System, Acceptance)	Complex	8
4.2.1	Execute All Tests	Complex	4
4.2.2	Record Results & Report Bugs	Medium	2
4.2.3	Final Validation & Regression Testing	Medium	2
4.3	User Guide & Deployment	Simple	4

Source: DineGODevelopment_Task_Management.xlsx [1]

1.2 Project Objectives

Overall Objective:

Develop a robust online restaurant table reservation system that meets high standards of quality, timeliness, and efficient resource allocation.

Specific quality targets:

The table below shows the software quality metrics applied in the project test plan. The metrics include test coverage, maximum allowable number of defects, defect rate, along with notes for each testing

phase. The goal is to ensure that the product quality meets the technical requirements and user expectations at all levels of testing.

Table 3: Software Quality Targets per Testing Plan.

#	Testing Stage	Test Coverage	No. of Defects	% of Defect	Notes
1	Reviewing	100%	<2	<2%	Internal code review
2	Unit Test	≥95%	<5	<3%	Per-module unit testing
3	Integration Test	≥90%	<5	<3%	Module integration
4	System Test	≥95%	<3	<2%	Full system testing
5	Acceptance Test	100%	0	0%	Must meet all requirements

Milestone Timelines (%): ≥ 90%

Allocated Effort (man-days): 5 members × 15 weeks × 2.5 days/week = 190 man-days (estimated; subject to actual project conditions).

1.3 Project Risks

Throughout the development process, the project may encounter various risks that can affect progress, quality, or overall success. Identifying potential risks early and preparing response strategies helps the team minimize negative impacts and ensure project stability.

The table below outlines common project risks, their potential impact, likelihood of occurrence, and corresponding response plans.

Table 4: Project Risk Assessment and Response Plan

#	Risk Description	Impact	Possibility	Response Plans
1	Lack of experience with new technologies	High	Medium	Internal training, knowledge sharing, mentoring
2	Sudden change in requirements from client/supervisor	Medium	Medium	Agile approach, feedback at every sprint
3	Delays due to academic workload or schedule clashes	High	High	Clear task division, weekly progress reporting
4	Integration issues across multiple modules	Medium	Medium	Continuous integration testing, strict code review
5	Data security breach/user data leakage	High	Low	Implement JWT/OAuth, regular security reviews
6	Insufficient server resources for deployment	Medium	Low	Use free cloud/server trials, code optimization
7	High bug count in the late project stage	High	Medium	Early testing, immediate bug fixing upon detection
8	Human resources risk	Medium	Medium	Cross-training, maintain clear documentation, early recruitment of backup personnel

2. Management Approach

2.1 Agile (Scrum) Workflow & Practices

The DineGO project will be managed using an Agile (Scrum) approach to ensure flexibility, transparency, and effective collaboration throughout the development process.

The project runs from May 12, 2025 to August 15, 2025, and is divided into five sprints, each lasting 2 weeks, following the standard Scrum framework. This cycle includes sprint planning, development, review, and retrospective phases.

The Sprint schedule is defined as follows:

Table 5: DineGO Agile Sprint Schedule and Key Development Activities

#	Sprint	Duration	Key Activities
1	1	12/05/2025 – 25/05/2025	System setup, requirement analysis, UI wireframe, and product backlog creation, UI wireframe & prototype, Database ERD & architecture design
2	2	26/05/2025 – 08/06/2025	Implement authentication module (register, login, role-based access), user profile management, restaurant management (Owner CRUD), basic database schema, API scaffolding
3	3	09/06/2025 – 22/06/2025	Develop reservation flow (create/view/cancel), real-time table availability view, menu & dish management (Owner CRUD), validation rules (capacity, no-show, large party)
4	4	23/06/2025 – 06/07/2025	Implement order & cart (add/update/remove items), integrate payment gateways (Casso, VietQR, MoMo sandbox), voucher and loyalty features, admin panel & reporting dashboard
5	5	07/07/2025 – 20/07/2025	Integrate AI assistant (recommendation, chatbot Q&A), develop social features (post, review, feedback, rating), implement notification & realtime chat (SignalR), finalize integration, conduct user testing
6	6	21/07/2025 – 15/08/2025	Regression testing, performance & security optimization, fix critical bugs, prepare deployment scripts, finalize documentation, deployment to production, presentation preparation

The team conducts daily stand-up meetings and uses the following tools to support Agile practices:

- **GitHub Projects** for sprint backlog and issue tracking
- **Google Sheets** for sprint planning and tracking contributions
- **Zalo** for real-time communication
- **Google Meet** for weekly reviews and retrospectives

This Agile process ensures that the project remains adaptable to changes while maintaining continuous delivery and incremental improvements throughout the development timeline.

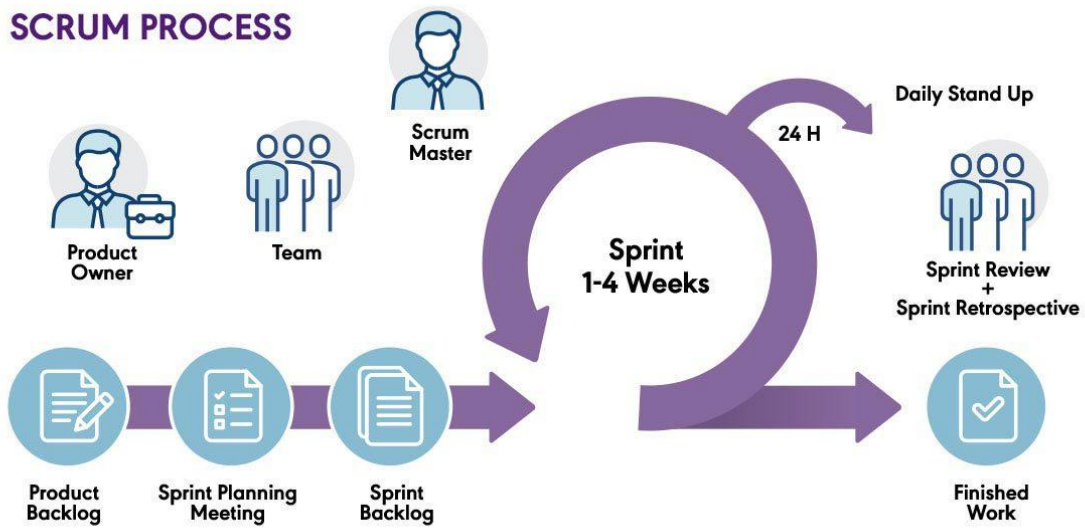


Figure 5: Software Development Lifecycle for DineGO[1]

- **Planning:** The team defines all required features and tasks, then divides work into manageable sprints (1–2 weeks per sprint). Tasks are prioritized and tracked using tools such as Trello or Google Sheets.
- **Roles & Responsibilities:** Each member is assigned specific roles (Leader, Developer, Tester, Reviewer, etc.), with regular progress reviews and support from all team members.
- **Communication:** Daily stand-up meetings (online/offline) are held to quickly update progress, resolve issues, and align on next steps. Team members communicate openly and promptly using messaging and video call tools.
- **Iteration & Feedback:** After each sprint, a sprint review and retrospective are conducted to assess what went well, what needs improvement, and to plan the next iteration. Continuous feedback from all stakeholders (team members, supervisor) is highly encouraged.
- **Documentation:** All important project documentation (requirements, designs, reports, meeting notes) is stored and regularly updated in a shared drive (Google Drive).
- **Quality Control:** All deliverables are subject to code review, testing, and team validation before moving to the next phase.
- **Risk Management:** Potential risks are identified early, and response plans are discussed and updated in each sprint.

This approach enables the team to adapt to changes, deliver incremental value, and ensure a high-quality and successful software product.

2.2 Team Roles & Collaboration Structure

To ensure a structured and effective implementation of the Agile Scrum methodology in the DineGO project, each team member is assigned a specific role within the Scrum framework. The responsibilities are clearly defined to align with best practices in Agile software development, aiming to maximize collaboration, accountability, and product value delivery.

[1] https://top20review.com/phuong-phap-agile/

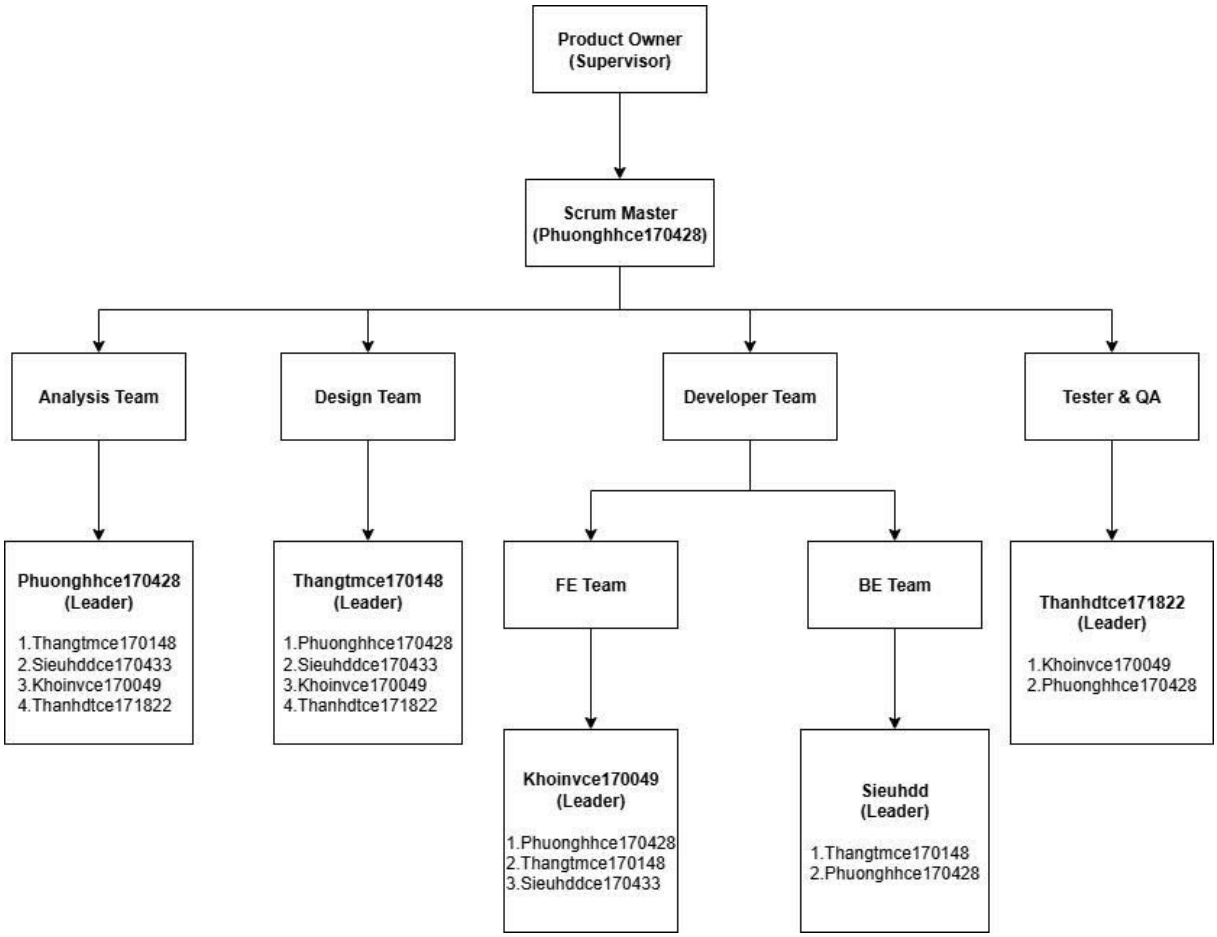


Figure 6: Team Roles and Collaboration Structure of the DineGO Project.

Product Owner (PO):

As the Product Owner (PO), this role serves as the primary liaison between stakeholders and the development team. The PO is responsible for articulating the project vision, optimizing the value delivered by the product, and ensuring that the team builds what is most valuable for the end users. Core responsibilities include:

- **Defining and Managing the Product Backlog**: He constructs and continuously refines the Product Backlog based on feedback from stakeholders, market demands, and development constraints. Each item is prioritized to maximize business value.
- **Clarifying Requirements and Sprint Objectives**: The PO communicates the purpose and expected outcomes of each sprint, ensuring the team clearly understands the scope and value of the backlog items.
- **Decision-Making Authority**: He makes critical decisions regarding product releases, trade-offs between scope and timeline, and adjustments to features based on evolving requirements.
- **Evaluation of Sprint Outcomes**: At the conclusion of each sprint, the PO participates in the Sprint Review to assess the deliverables against acceptance criteria and align future priorities.

Scrum Master:
In the role of Scrum Master, the individual is entrusted with facilitating the Scrum process and ensuring that Agile principles are properly applied throughout the project. Acting as a servant-leader, the Scrum Master supports both the development team and the Product Owner. Key responsibilities include:

- **Agile Coaching and Facilitation**: Guiding the team in the adoption and implementation of Scrum practices, promoting self-organization, and empowering team members.
- **Organizing Scrum Events**: Planning and moderating all Scrum ceremonies such as Daily Stand-Ups, Sprint Planning, Sprint Review, and Sprint Retrospective.
- **Removing Impediments**: Identifying and addressing obstacles that hinder the team's progress, including both internal conflicts and external dependencies.
- **Promoting Continuous Improvement**: Encouraging a mindset of agility and adaptability through retrospectives and team feedback mechanisms.

Development Team:
The Development Team is a cross-functional, self-organizing unit responsible for delivering working increments of the software at the end of each sprint. It includes both Developers and a dedicated Tester, all of whom collaborate closely to ensure product quality and technical excellence.

Developers:
The developers are responsible for translating the product vision into functional software. Their tasks are not limited to coding but encompass the entire development lifecycle. Their responsibilities are as follows:

- **Feature Implementation**: Transform Product Backlog items into fully functional features that meet the Definition of Done and align with sprint objectives.
- **Code Quality and Standards**: Adhere to predefined coding standards, perform code refactoring when necessary, and utilize version control systems effectively.
- **Collaboration and Code Reviews**: Conduct peer reviews to maintain code integrity and consistency, while also contributing to collaborative problem-solving.
- **Automation and Integration**: Apply CI/CD practices where feasible to enhance deployment efficiency and reduce manual errors.

Testers (QA):
The Tester plays a critical role in verifying that the software meets quality standards and business requirements. His role ensures that functional increments are reliable, stable, and meet end-user expectations. Responsibilities include:

- **Test Planning and Execution**: Design comprehensive test cases based on user stories and acceptance criteria, and execute manual or automated testing procedures.
- **Quality Assurance Activities**: Perform integration testing, system testing, and regression testing across builds to ensure consistency and functionality.
- **Defect Management**: Identify, log, categorize, and track defects using appropriate tools. Collaborate with developers to reproduce and resolve issues.
- **Sprint Validation**: Participate in Sprint Reviews by validating feature functionality and usability from the end-user perspective.

2.3 Project Process

The DineGO project adopts the Agile (Scrum) software development methodology to ensure flexibility, transparency, and iterative delivery throughout the development lifecycle. This approach fosters continuous feedback, rapid adaptation to changes, and strong collaboration among team members.

- **Iteration:** The project timeline is divided into short, time-boxed iterations known as **Sprints**, each lasting from one to two weeks. Within each Sprint, the team aims to deliver a functional product increment that contributes to the overall system.
- **Workflow:** Each Sprint begins with a Sprint Planning Meeting, where the team selects a set of prioritized Product Backlog items to implement. Throughout the Sprint, the team engages in daily development and testing activities. At the end of the Sprint, a Sprint Review is conducted to demonstrate the completed work, followed by a Sprint Retrospective to reflect on team performance and identify improvement opportunities.
- **Team Roles:** In the Scrum framework adopted by the DineGO project, each team member assumes a clearly defined role that contributes to the overall agility and efficiency of the development process. The Product Owner functions as the central figure responsible for defining the vision of the product, maintaining and prioritizing the product backlog, and ensuring that the development outcomes align with stakeholder expectations. This role is vital in bridging the gap between the technical team and the business goals, guiding the team toward delivering maximum value in each sprint.
- **Daily Standup:** To maintain transparency and alignment, the team holds **Daily Stand-Ups**, typically lasting 15 minutes. Each member shares their progress, identifies any impediments, and outlines planned tasks for the day. This practice ensures early detection of issues and enhances team coordination.
- **Backlog Management:** All Product Backlog items and Sprint tasks are managed using tools such as Trello, GitLab Boards, or Google Sheets, depending on the team's operational needs. These tools facilitate task assignment, priority tracking, progress visualization, and documentation of requirements and deliverables.

Process Steps:

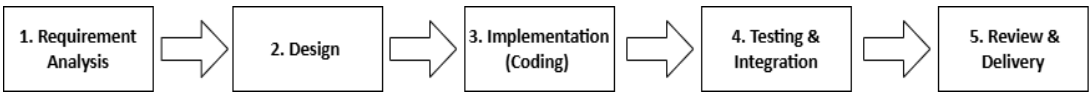


Figure 7: Software Development Lifecycle for DineGO

Advantages of Agile:

The adoption of Agile methodology offers several significant benefits that align with the dynamic nature of modern software development. One of the most notable advantages is the ability to deliver functional software incrementally and continuously from the early stages of the project. This approach ensures that stakeholders are consistently presented with usable outcomes, enhancing transparency and engagement throughout the development lifecycle.

Furthermore, Agile promotes a high degree of responsiveness to change. Rather than rigidly adhering to predefined plans, the process encourages iterative refinement based on evolving requirements and stakeholder input. This flexibility enables the team to align more closely with actual user needs and shifting project priorities.

Another key strength of Agile lies in its emphasis on collaboration and open communication. Frequent interactions among developers, testers, and product stakeholders foster a shared understanding of objectives and challenges, leading to better decision-making and collective ownership of outcomes.

Finally, the iterative nature of Agile facilitates early detection and timely resolution of defects. Through continuous integration, regular testing, and retrospective evaluations, quality issues are identified and addressed swiftly, thereby reducing technical debt and improving overall system stability.

2.4 Quality Management

To ensure high-quality software that meets user needs and business requirements, the team applies the following quality management approaches:

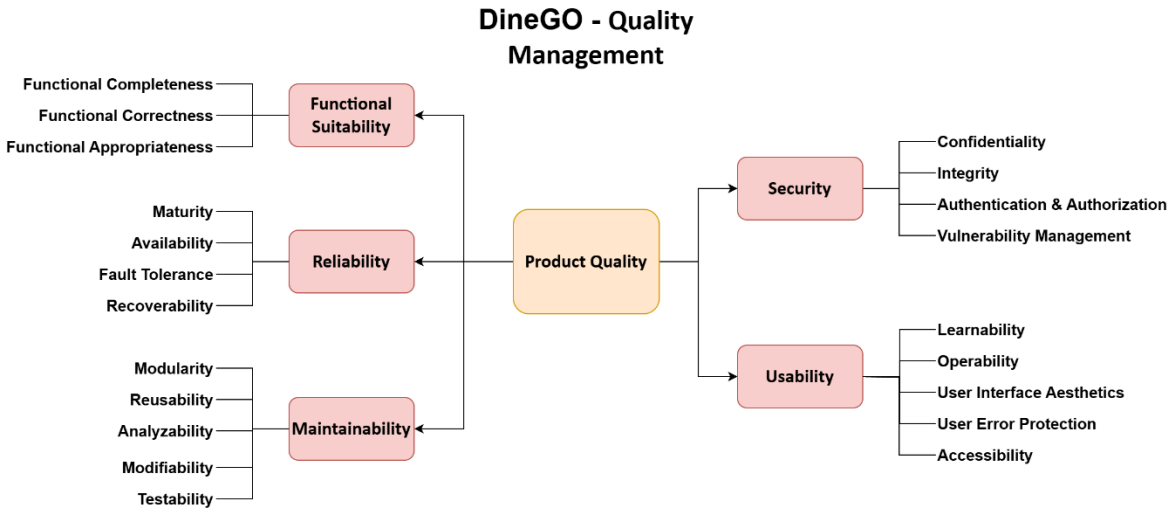


Figure 8: Selected Software Product Quality Characteristics

- **Functional Suitability:** The system is designed with a strong emphasis on fulfilling user requirements with precision and consistency. All functional specifications are clearly defined and documented prior to implementation, ensuring traceability between requirements and developed features. Continuous validation sessions with stakeholders are conducted to confirm that each deliverable aligns with its intended functionality and satisfies end-user expectations.
- **Reliability:** To guarantee dependable system behaviour, comprehensive unit and integration testing are employed early and consistently throughout the development process. Automated test suites are integrated into a continuous integration (CI) pipeline, allowing for real-time defect detection with every code commit. In addition, full system testing is performed prior to each major release to validate overall stability under production-like conditions.
- **Maintainability:** Code quality is maintained through strict adherence to industry-standard coding conventions and mandatory peer reviews for all significant code changes. The project enforces modular design principles, resulting in a codebase that is both readable and adaptable. Periodic refactoring cycles are conducted to address technical debt and to accommodate evolving project requirements.
- **Security:** Security is embedded into the development process from the ground up. Secure coding practices are adopted to mitigate common vulnerabilities, and proper handling of sensitive data is ensured through the implementation of robust authentication and

authorization protocols. Regular security audits and reviews are performed to assess potential risks and ensure the integrity of the system.

- **Usability:** User experience is considered a critical success factor. Interfaces are designed based on direct user feedback and established usability standards. Wireframes and prototypes are iteratively refined through user testing, and formal user acceptance testing (UAT) is conducted to verify interface intuitiveness and satisfaction prior to final deployment.

The overall testing strategy combines multiple levels of validation into a unified, continuous quality assurance workflow. Rather than treating testing as a distinct phase, it is fully integrated into the development process. Developers are responsible for writing and maintaining unit tests as they build functional modules, while integration and system testing are executed in parallel to ensure that components interact correctly and perform reliably. Finally, acceptance testing is performed in collaboration with stakeholders to confirm that completed features fulfil business requirements and are ready for deployment. This holistic approach allows the team to maintain a high level of software quality while remaining responsive to feedback and change.

Table 6: Additional Support Roles (optional, depending on team structure)

Role	Description
UI/UX Designer	Create wireframes/prototypes; optimize usability and visual flow.
Release Manager	Plan and track release cycles, manage release readiness and approvals.
Document Controller	Ensure proper documentation organization, version control, and permissions.
DevOps Engineer	Maintain CI/CD pipelines, automate deployments, manage environments.

Quality Objectives:

- Minimize defects through preventive measures and comprehensive testing.
- Maximize code coverage and ensure all functionalities are tested.
- Ensure system stability, maintainability, and security.
- Deliver software that fulfills user needs, is easy to maintain, and can be deployed with confidence.

2.5 Training Plan

To address any skill gaps within the team:

- **Onboarding:** New members are guided through the project's codebase, architecture, and workflow tools.
- **Skill Training:** Short sessions and knowledge sharing on technologies such as ASP.NET Core, SQL Server, Git, and Agile practices.
- **Peer Learning:** Team members support each other by sharing expertise (pair programming, code review).
- **External Resources:** When needed, the team utilizes official documentation, online courses, and community forums.

Table 7: Training & Development Plan

Training Area	Participants	When, Duration	Waiver Criteria
Agile Methodology	All members	1 week	Previous Agile experience
ASP.NET Core	Developers	2 weeks	Completed sample project
SQL Server	Developers	1 week	Passed sample queries
Git & Version Control	All members	2 days	Pass Git workflow checklist
AI Fundamentals	Interested team members	1 week	Completed AI introductory exercises

3. Project Deliverables

To ensure project tracking and timely submission, each major document deliverable is scheduled with a clear due date and purpose. These deliverables cover all essential stages of the software development process.

The following table outlines the project deliverables, their respective deadlines, and the key contents of each document.

Table 8: Project Deliverables Schedule

#	Deliverable	Due Date	Notes
1	Project Introduction Report	15/05/2025	Overview, team, background
2	Project Management Plan	17/05/2025	Scope, estimation, risks
3	Software Requirement Spec.	24/05/2025	User & system requirements
4	Software Design Document	31/06/2025	Architecture, database, diagrams
5	Software Test Documentation	25/07/2025	Test plan, cases, reports
6	Software User Guides	07/08/2025	Install/user manual
7	Final Project Report	15/08/2025	Full deliverable, results

4. Responsibility Assignments

To ensure work is distributed effectively and each member clearly understands their role, the team has defined a Responsibility Assignment Matrix (RAM). This matrix shows who is primarily responsible (Do), who supports, who reviews the work, and who should be informed.

The following table outlines responsibilities for each task in the project and assigns them to team members accordingly.

D~Do; R~Review; S~Support; I~Informed; <blank>- Omitted

Table 9: Responsibility Assignment Matrix (RAM)

Responsibility	Phuonghh	Sieuhdd	Khoinv	Thangtm	Thanhdt
Create Reports (1–7), Review & Finalize Docs	D	R	R	S	R
Project Schedule & WBS Management	S	S	D	R	I
ERD Database, Table Schema, Normalization	R	D	S	S	R
System Architecture & Diagrams (Component, Deployment, Package)	D	R	S	S	R
Build Core Business Logic (Reservation, Ordering, Payment)	D	S	R	S	S
Implement UI & Controllers	R	D	S	S	R
API/Service Layer	D	S	S	S	R
Integration of Modules (Frontend–Backend, External Services)	D	R	R	S	R
Debugging & Refactoring	S	S	D	R	I
Create Test Cases (Unit, Integration, Acceptance)	R	D	S	S	R
Execute Tests & Record Results	D	R	S	S	R
Bug Tracking & Regression Testing	D	S	R	S	S
VPS/AWS Setup, Deployment	R	D	S	S	R
User Guide & Training Materials	D	S	S	S	R
Reservation Management	D	R	R	S	R
Ordering & Payment (Bank Transfer, VietQR, MoMo, COD)	S	S	D	R	I

Social Interaction (Post, Blog, Review, Rating, Chat)	R	D	S	S	R
AI Assistant (Recommendation, Chatbot)	D	R	S	S	R
Admin Management (License Approval, Vouchers, Notifications, System Log)	D	S	R	S	S

5. Project Communications

Effective communication is crucial to ensure collaboration, timely updates, and issue resolution within the team. The following communication plan defines how and when the team will interact, as well as which tools will be used.

The table below outlines the communication channels, responsible parties, purpose, schedule, and tools/methods applied.

Table 10: Project Communication Plan

Communication Item	Who/ Target	Purpose	When, Frequency	Type, Tool, Method(s)
Weekly Team Meeting	All team members	Progress update, planning	Every Friday	Google Meet, in-person
Sprint Review	Team, Supervisor	Review deliverables	End of each Sprint	Google Meet
Documentation Sync	All team members	Update shared docs	Continuous	Google Drive, Docs/Sheets
Code Review	Dev team	Check/merge code changes	Before merge	Github Pull Request
Urgent Issues	All team members	Quick coordination	As needed	Messenger, Zalo, Email

6. Configuration Management

6.1 Document Management

To ensure consistency and traceability throughout the project lifecycle, all project-related documents are stored and managed in a centralized Google Drive workspace named Docs_DineGO. This folder is structured into categorized subfolders to support ease of access, version control, and collaboration.

The screenshot below illustrates the organization of our document management system.

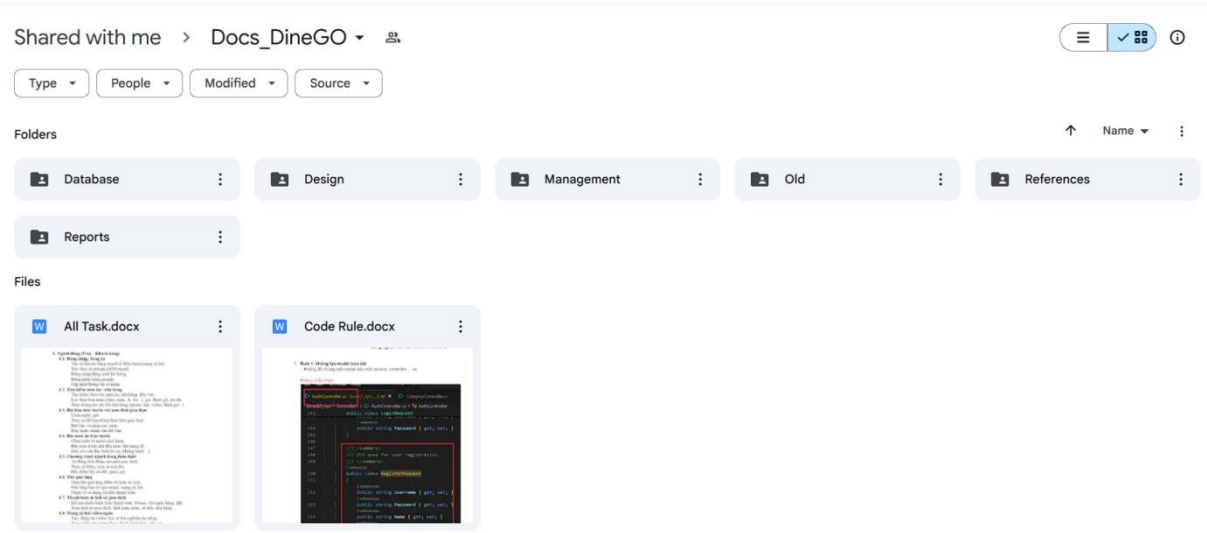


Figure 9: Document Repository Structure on Google Drive

All project documents are systematically stored on Google Drive following a clearly defined folder structure. The main directories include:

- **Reports**: This directory contains all official project reports and deliverables, arranged in chronological order corresponding to the project milestones. These documents include the Project Introduction, Management Plan, Software Requirement Specification, Software Design Document, Test Documentation, User Guide, and Final Report. Each file is labeled with version numbers and submission dates, ensuring that both the team and academic supervisors can easily track the project's progress over time.
- **Management**: The Management folder houses essential planning and coordination documents that govern the project's execution. Key artifacts include the Work Breakdown Structure (WBS), Gantt charts, detailed effort estimations, and weekly tracking logs. These documents serve as the foundation for resource allocation, timeline monitoring, and risk management. The folder is regularly updated to reflect changes in scope, task assignments, or scheduling, and is used extensively during sprint reviews and supervisor meetings.
- **Design**: This directory contains all design-related artifacts developed during the system architecture and software modeling phases. It includes Entity Relationship Diagrams (ERDs), database schema scripts, UI wireframes, use case diagrams, screen flow diagrams, and interaction models. These materials provide a visual and structural representation of the

system, bridging the gap between functional requirements and implementation. The content of this folder plays a critical role in design validation and serves as a technical reference throughout the development process.

- **Reference**: The Reference folder includes academic and practical materials consulted during the planning and execution of the project. These may consist of sample capstone reports, institutional guidelines, lecture notes, and industry-standard documentation templates. This repository is intended to support research, ensure alignment with expected deliverable formats, and provide examples for best practices in project structure and reporting.

Access rights are assigned based on member roles and responsibilities. For example, project managers may have editing rights for management documents, while general members are granted view-only access. All changes are version-controlled through Google Docs/Sheets' built-in revision history.

6.2 Source Code Management

To manage source code efficiently and ensure team collaboration, the DineGO project follows a Git-based branching strategy on GitHub. This approach supports separation of development, testing, and production environments, and provides flexibility for feature enhancements, bug fixes, and code refactoring.

The diagram below illustrates the overall workflow for managing source code in the project.

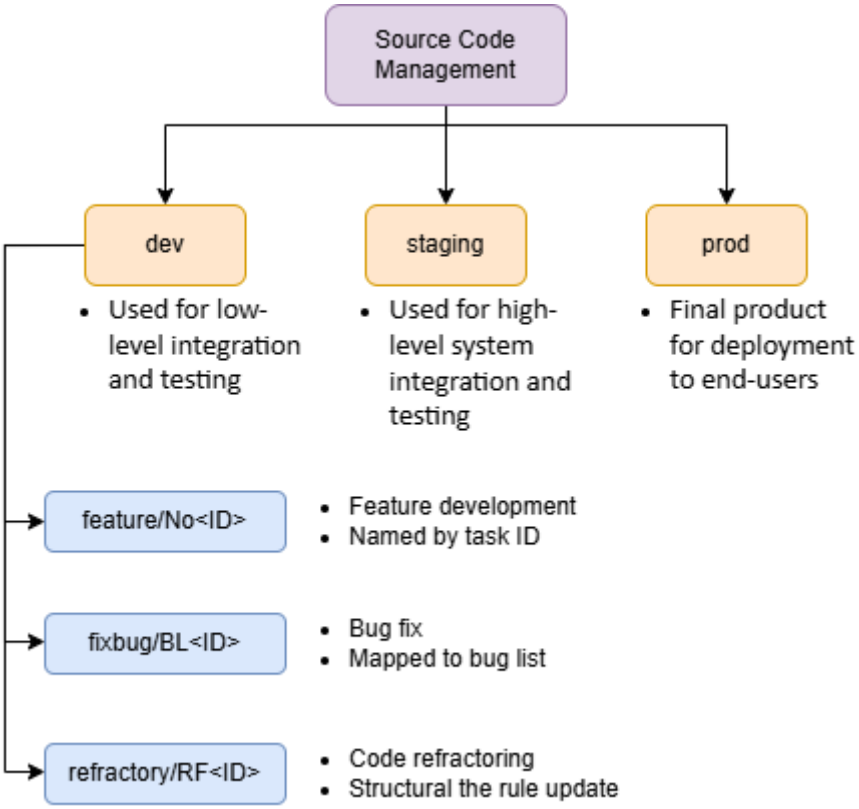


Figure 10: Source Code Management Workflow of DineGO Project

Our version control process is implemented on GitHub with the following structure:

- **Branch Naming Convention**: Each task assigned is associated with a unique ID from the WBS. Branches are named using the format **No<ID>**, corresponding to the task ID.
- **Code Review Workflow**:
  - Team members create branches for their tasks.
  - Each Pull Request (PR) must include the task ID and link back to its WBS item.
  - Before merging, the PM or Tech Lead will review the PR. Feedback, if any, will be left in comments directly on GitHub.
  - If the PR meets all criteria, it will be approved and merged into the dev branch.
- **Code Quality Standards**:
  - All code must comply with the standards outlined in the **Code_rule** document.
  - Team members must check compliance before submitting code for review.
- **Branch Environments**:
  - **dev**: Used for low-level integration and testing (unit tests, integration tests)
  - **staging**: Used for high-level system integration and testing (system tests)
  - **prod**: Final product for deployment to end-users (acceptance testing)
- **Branch Naming Convention**:
  - **feature/No<ID>**: For feature development tasks aligned with the WBS task ID.
  - **fixbug/BL<ID>**: For bug fix tasks, mapped to the BugList sheet in the WBS.
  - **refractory/RF<ID>**: For code refactoring or structural rule updates, mapped to the Change Requirement sheet.

Task Mapping: Each GitHub branch must include the corresponding task ID. Pull Requests (PRs) are linked to their task and submitted for review. PMs or Tech Leads are responsible for reviewing and merging.

6.3 Tools & Infrastructures

To support development, collaboration, deployment, and documentation, the DineGO project utilizes a variety of tools and infrastructure services. These tools were selected based on compatibility, team familiarity, and project requirements.

The following table provides an overview of the tools and platforms used in the project.

Table 11: Tools and Infrastructure Overview

Category	Tools / Infrastructure
Technology	ASP .NET, Python
Database	SQL Server 2022, Navicat Data Modeler 4
IDEs/Editors	Visual Studio Code Community
Diagramming	StarUML, DrawIO
Documentation	Google Docs/Sheets/Slides
Version Control	GitHub (Source Codes), Google Drive (Documents), Fork, Git Desktop
Deployment Server	Virtual Private Server (VPS)
AI Platform	Hugging Face
Project Management	Google Sheets (Schedule, Tasks), GitHub (Tasks, Defects), office 365
Cloud Service	Amazon Web Service(S3 Bucket)
Geolocation	Open Street Map
Payment Integration	Casso, VietQR

6.3.1 ASP .NET

ASP .NET serves as the core backend framework for the DineGO system. Developed and supported by Microsoft, it is a high-performance, open-source platform tailored for building modern, scalable, and secure web applications. The project utilizes the Model-View-Controller (MVC) architecture to separate presentation logic from business logic and data access, thereby enhancing maintainability and testability. Additionally, ASP .NET facilitates the development of RESTful APIs, ensures seamless integration with SQL Server, and supports advanced authentication and authorization mechanisms, making it an ideal choice for enterprise-grade systems like DineGO.

Figure 11: ASP.NET logo representing the chosen backend framework[2]

6.3.2 SQL Server 2022 & Navicat Data Modeler 4

SQL Server 2022 is employed as the primary relational database management system (RDBMS) for DineGO. It is responsible for storing, organizing, and retrieving structured data such as user accounts, reservations, restaurant profiles, and transactional records. To complement this, Navicat Data Modeler 4 is used to design and manage Entity Relationship Diagrams (ERDs), enabling the development team to visually define table relationships, enforce data integrity constraints, and document database

[2] https://www.blendhosting.com/development/choosing-asp-net-hosting/

schemas effectively. Together, these tools facilitate a robust and well-structured data layer that ensures system reliability and scalability.

Figure 12: Microsoft SQL Server 2022 logo – database management system[3]

6.3.3 Visual Studio Code Community

Visual Studio Code (VS Code) is the integrated development environment (IDE) selected for source code authoring across both frontend and backend components. Its lightweight architecture, combined with an extensive extension marketplace, allows for rich development capabilities including syntax highlighting, intelligent code completion, debugging tools, and Git integration. VS Code supports multiple programming languages used in the project such as C#, HTML, JavaScript, and CSS. Its flexibility and productivity-enhancing features make it a highly effective tool for team collaboration and agile software development practices.

Figure 13: Visual Studio Code logo – primary code editor for development[4]

6.3.4 StarUML & DrawIO

For system modeling and visual documentation, the team utilizes StarUML and DrawIO. StarUML is adopted for designing software artifacts through standardized UML diagrams, including Use Case, Class, and Sequence diagrams. These diagrams are crucial for illustrating system behavior, structural organization, and user interactions in a way that aligns with both academic and professional software engineering practices. DrawIO (also known as diagrams.net) is employed for quick prototyping of UI wireframes, ERDs, and deployment diagrams due to its intuitive drag-and-drop interface and cloud-based integration. These tools collectively support architectural clarity and effective communication with both technical and non-technical stakeholders.

Figure 14: Draw.io logo – diagramming and modeling tool[5]

6.3.5 Google Docs, Sheets, and Slides

The DineGO project extensively employs the Google Workspace suite—specifically Google Docs, Sheets, and Slides—for collaborative documentation, planning, and presentation purposes. Google Docs serves as the primary platform for drafting and maintaining project deliverables such as

[3] https://www.tmssoftware.com/site/tmswebcorevsc.asp?r=popup
[4] https://www.tmssoftware.com/site/tmswebcorevsc.asp?r=popup
[5] https://alternative.me/draw-io

requirement specifications, design documents, and progress reports. Google Sheets facilitates task tracking, scheduling, and effort estimation through spreadsheets shared in real-time among team members. Google Slides is used to prepare visual content for stakeholder presentations and final defense.

Figure 15: Google Workspace icons – Docs, Sheets, Slides, and Forms[6]

6.3.6 GitHub, Fork & GitHub Desktop

For source code version control, issue tracking, and collaborative development, the team adopts GitHub as the central platform. GitHub enables structured management of repositories with features such as pull requests, code reviews, branching strategies, and release tagging. The project follows a GitFlow-inspired workflow, using feature branches for new functionality, hotfix branches for urgent issues, and staging/production branches for deployment readiness. Fork and GitHub Desktop are supplementary tools used by team members preferring graphical interfaces over command-line Git, thus lowering the barrier to version control participation.

Figure 16: GitHub logo – version control and collaboration platform[7]

6.3.7 Virtual Private Server (VPS)

To host and deploy the DineGO web application in a real-world simulation environment, the team utilizes a Virtual Private Server (VPS). The VPS provides root-level access, allowing configuration of runtime environments such as the .NET Core framework, SQL Server drivers, and reverse proxies like Nginx. It serves as the staging and production environment for continuous integration and final deployment.

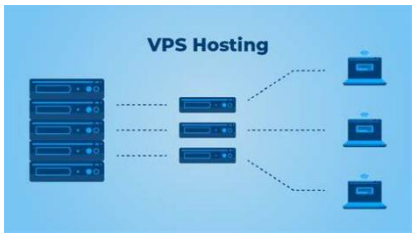


Figure 17: VPS Hosting illustration – deployment infrastructure[8]

6.3.8 Project Management Tools

Effective task management and progress tracking are achieved through a combination of Google Sheets and GitHub's issue board. Google Sheets is used to maintain the project schedule, allocate team

[6] https://howtotechnaija.com/how-to-export-google-docs-to-pdf-excel-word-text/
[7] https://icon-icons.com/icon/github-logo/168170
[8] https://ototekno.id/pengertian-virtual-private-server-vps/

responsibilities, and log weekly activity in accordance with academic reporting standards. GitHub Issues and Projects are used to manage technical tasks, defects, and development milestones following Agile methodology. This dual-system approach enables both managerial oversight and technical control, ensuring alignment with both academic requirements and professional software engineering practices. Office 365 tools are occasionally utilized for compatibility in document formatting and official submissions.

6.3.9 AI & Machine Learning Tools

Description: The DineGO system integrates machine learning models to predict food-related tags and generate context-based dish suggestions. These models are trained and deployed using Python, leveraging the Hugging Face Transformers library.

- **AI Language & Frameworks**: Python 3.10, Hugging Face Transformers, PyTorch
- **Training Environment**: Google Colab (GPU T4/A100) and local machine (NVIDIA GPU)
- **Model Management**: Stored locally (../models) and backed up to Google Drive
- **Inference Deployment**: .NET Core API invokes Python scripts via process execution (e.g., predict_unified.py, predict_response.py)
- **Objectives**:
  - [9]**DistilBERT**: Multi-label classification to assign food tags from user descriptions
  - [10]**VietT5**: Context-aware text generation for natural food suggestions
- **Benefits**: Personalizes dish recommendations by combining behavioral analysis with natural language responses

Figure 18: Hugging Face logo – AI/ML model hosting and sharing platform[11]

6.3.10 Python

Python is utilized in the system as a versatile and powerful programming language, primarily for implementing AI-related functionalities and backend automation scripts. In the DineGO system, Python plays a crucial role in developing machine learning models for tasks such as food tag prediction and natural language processing for the AI chatbot. Leveraging Python's rich ecosystem of libraries—such as TensorFlow, PyTorch, and Transformers—the development team can efficiently train, evaluate, and deploy AI models that enhance user experience and provide intelligent recommendations.

Additionally, Python is used for data preprocessing, integration testing of AI services, and creating automation scripts that interact with external APIs. Its readability and extensive community support make it ideal for rapid prototyping and integrating complex AI algorithms into the main system.

[9] Hugging Face – distilbert-base-uncased
[10] Hugging Face – VietAI/vit5-base
[11] https://www.turingpost.com/p/huggingfacechronicle

Figure 19: Python logo – programming language for AI model development[12]

6.3.11 AWS - S3

Amazon Simple Storage Service (AWS S3) is utilized in the system as an object storage service to manage and store images uploaded by users and restaurants. When a user or a restaurant uploads an image (e.g., profile picture, food photo, menu image), the system sends a secure API request to directly upload the file to S3, ensuring high-speed transfer and reliability. AWS S3 offers scalable storage capacity, secure data retention with multiple layers of protection, and flexible access control—either through public URLs or restricted permissions based on the application's requirements.

Figure 20: Amazon S3 logo – cloud storage service[13]

6.3.12 Casso (Payment)

Casso is integrated into the system as an automated payment connection platform, enabling the retrieval of transaction lists from linked bank accounts. The primary function of Casso in the system is to obtain the latest payment records to verify and confirm customers' payment statuses. When a customer makes a payment (e.g., a table reservation fee), the system checks the payment detail via Casso's API, including transaction amount, timestamp, and transfer description.

Figure 21: Casso logo – automated payment processing platform integrated in DineGO[14]

6.3.13 Open Sheet Map

OpenStreetMap (OSM) is integrated into the system as an open-source mapping platform that provides geolocation and map rendering services. The primary function of OSM in the system is to display interactive maps for restaurant locations, enabling users to easily search for and navigate to their desired destinations. By leveraging OSM's extensive geographical database, the system can show

[12] https://www.python.org/
[13] https://www.next.inc/uploads/2023/12/amazon_s3-750x422.jpg
[14] https://casso.vn/wp-content/uploads/2023/07/CASSO_LOGO_FINAL-01-600x600.jpg

detailed street maps, landmarks, and area boundaries without relying on proprietary mapping solutions.

In the DineGO system, OSM is used to visualize the restaurant's position, user's current location, and surrounding points of interest. It supports features such as zooming, panning, and selecting a specific location on the map for reservation purposes. The integration of OSM ensures cost-effective, customizable, and reliable map services, making it possible to maintain flexibility in map design and functionality while avoiding licensing restrictions from commercial providers.

Disclaimer: The map data displayed via OpenStreetMap may contain disputed or controversial boundaries. The system does not endorse or support any territorial claims, and the map is used solely for navigation and restaurant location purposes.

Figure 22: OpenStreetMap logo – open-source mapping[15]

6.3.14 VietQr

VietQR is integrated into the system as a solution to generate QR codes for quick payment according to Napas 247 standards. This technology allows users to scan QR codes with banking applications or e-wallets to make fast and secure money transfers.

In DineGO, VietQR is used to support payment of table reservations and other transactions. When customers create a table reservation, the system will generate a VietQR code containing full recipient information, amount and transaction content. This helps to minimize errors when entering information manually and shorten payment time.

VietQR brings convenience, security and wide compatibility with many banks and e-wallets in Vietnam, contributing to improving user experience during the table reservation and payment process.

Figure 23: VietQR logo – QR payment solution in Vietnam[16]

[15] https://www.openstreetmap.org/
[16] https://vietqr.net/

III. Software Requirement Specification

1. Product Overview

1.1 Overview

DineGO is a web-based food ordering and reservation system designed to streamline the restaurant service experience for customers, restaurant owners, and administrators. The system supports key features such as restaurant search by location, viewing menus, placing orders, booking tables, and managing order history. Customers interact with the platform through a user-friendly web interface, while restaurant owners manage their offerings via a dedicated dashboard. The diagram below presents the context of the system, showing the external entities (users, restaurant staff, administrators) and their interactions with the core system functionalities. DineGO is expected to evolve over future releases, integrating third-party payment gateways, delivery tracking services, and advanced recommendation systems to enhance user experience.

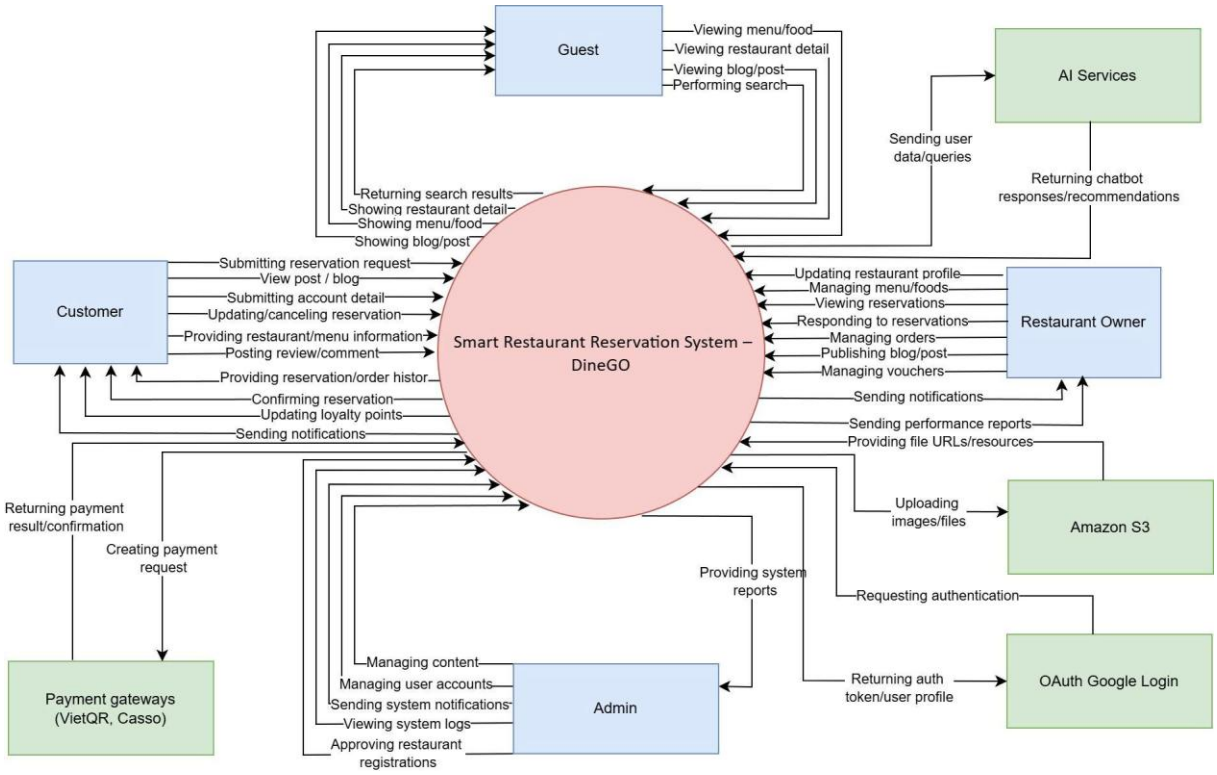


Figure 24: Context Diagram of the Smart Restaurant Reservation System – DineGO

1.2 Business Rules

To ensure consistent system behavior and enforce operational policies across all user roles, the DineGO platform defines a set of business rules. These rules govern core functionalities such as reservations, ordering, payment processing, loyalty programs, and platform usage. Clearly outlining these rules helps maintain system integrity, user trust, and compliance with both technical and business expectations.

Table 12*: Business Rules for the DineGO System*

ID	Rule Definition	Group
BR-01	The restaurant owner must set up complete restaurant information (name, location, opening hours, closing hours).	Reservation Rules
BR-02	The customer is only allowed to make a reservation at the latest 2 hours before the restaurant's closing time.	Reservation Rules
BR-03	The system must arrange reservation slots with a 30-minute interval.	Reservation Rules
BR-04	The customer is only allowed to make a reservation up to 7 days from the booking date.	Reservation Rules
BR-05	The system must prevent new reservations within 2 hours before or after an existing reservation for the same table.	Reservation Rules
BR-06	The customer must provide a complete email and phone number when making a reservation.	Reservation Rules
BR-07	The customer must make a reservation at least 30 minutes before the selected dining time.	Reservation Rules
BR-08	The customer must pay a deposit as specified by the restaurant.	Reservation Rules
BR-09	The system only accepts reservation statuses: Pending, Accepted, Canceled, Show, No-Show, Completed.	Reservation Rules
BR-10	The system must initialize a Reservation in the Pending status and automatically cancel it if not confirmed/payment is not completed within 10 minutes.	Reservation Rules
BR-11	The customer is not allowed to create a new Reservation if they have an existing Reservation in Pending status.	Reservation Rules
BR-12	The restaurant owner can mark a customer as "No-show" if they are 30 minutes late for the reservation time.	Reservation Rules
BR-13	The system must send a cancellation notification when a customer's Reservation is marked as "No-show".	Reservation Rules
BR-14	The system only allows the restaurant owner to change a Reservation status to Completed when the customer is present.	Reservation Rules
BR-15	The system must send a message to both the customer and the restaurant owner when a reservation is successful.	Reservation Rules
BR-16	The system must send a reminder notification to the customer 30 minutes before the reservation time.	Reservation Rules

ID	Rule Definition	Group
BR-17	The customer can only pay the deposit via VietQR; no other payment methods are allowed.	Reservation Rules
BR-18	The customer is only allowed to add items from the same restaurant to the cart; if they add items from a different restaurant, they must confirm deleting the old cart.	Ordering & Cart Rules
BR-19	The customer is only allowed to pay for an order via COD (Cash on Delivery).	Ordering & Cart Rules
BR-20	The customer must place an order with a minimum value of 200,000 VND and a maximum of 5,000,000 VND (by default); the restaurant owner can configure a different maximum but not exceeding 100,000,000 VND.	Ordering & Cart Rules
BR-21	The customer is only allowed to order a maximum of 50 units for one item; the restaurant owner can configure up to 200 units.	Ordering & Cart Rules
BR-22	The customer must provide their name, phone number, and delivery address before placing an order.	Ordering & Cart Rules
BR-23	The system must fix the item price at the time of order creation; subsequent changes will not affect the current order.	Ordering & Cart Rules
BR-24	The customer is only allowed to place a delivery order within a 10 km radius.	Ordering & Cart Rules
BR-25	The customer is not allowed to place an order 1 hour before the restaurant's closing time.	Ordering & Cart Rules
BR-26	The system must flag a customer as spam if a delivery fails due to a customer's fault; after 3 times, the account must be locked.	Ordering & Cart Rules
BR-27	The customer is only allowed to apply a maximum of 1 voucher per order.	Ordering & Cart Rules
BR-28	The system must add 1 point for every 10,000 VND the customer pays (after discount, before tax).	Loyalty & Voucher Rules
BR-29	The system must delete reward points after 365 days from the last interaction or according to policy.	Loyalty & Voucher Rules
BR-30	The customer can redeem points for a voucher, e.g., 100 points = 10,000 VND (according to the program configuration).	Loyalty & Voucher Rules
BR-31	The system must ensure that each voucher can only be used once and cannot be redeemed for cash.	Loyalty & Voucher Rules
BR-32	The Admin must set a maximum discount amount when creating a percentage-based voucher.	Loyalty & Voucher Rules
BR-33	The system does not allow a fixed-value discount voucher to exceed 50% of the order value.	Loyalty & Voucher Rules

ID	Rule Definition	Group
BR-34	The customer can only gift a voucher to another customer if they are friends.	Loyalty & Voucher Rules
BR-35	The customer can rate the restaurant with stars (1-5) and a feedback message.	Feedback Rules
BR-36	The system must calculate the average rating and round it to 1 decimal place.	Feedback Rules
BR-37	The customer is only allowed to leave a review after completing at least 1 order at the restaurant.	Feedback Rules
BR-38	The customer is only allowed to create a Post with content accompanied by a photo or video.	Post Rules
BR-39	The customer is only allowed to post a maximum of 10 photos (*.jpg, *.jpeg, *.png formats, size ≤ 5MB/photo).	Post Rules
BR-40	The customer is only allowed to post a maximum of 1 video (*.mp4, *.mov formats, size ≤ 100MB/video).	Post Rules
BR-41	The customer is only allowed to edit or delete a Post created by themselves.	Post Rules
BR-42	All Posts created by customers must be reviewed (approved/rejected) by an Admin before being publicly displayed.	Post Rules
BR-43	The Admin has the right to change a Post's status (approved, rejected) at any time.	Post Rules
BR-44	The restaurant owner must declare a prep_time (in minutes) for each item for the system to calculate the estimated preparation/delivery time.	Inventory & Menu Rules
BR-45	The restaurant owner must mark an item as "Out of stock" when it is no longer available; the system will not allow ordering but will still display it with a label.	Inventory & Menu Rules
BR-46	The system only stores the user's location and allows searching for items on a map when the customer accepts Cookies.	Inventory & Menu Rules
BR-47	The system must calculate the distance using the Haversine algorithm and sort restaurants from nearest to farthest.	Inventory & Menu Rules
BR-48	The system must store Audit logs (important changes) for at least 6 months and allow exporting them as CSV/PDF.	Account & Security Rules
BR-49	The customer must accept the Terms & Privacy Policy when registering.	Account & Security Rules
BR-50	The customer must verify their email to activate the account.	Account & Security Rules
BR-51	The system only allows OTPs to be valid for a maximum of 1 minute.	Account & Security Rules

ID	Rule Definition	Group
BR-52	The system must encrypt all passwords using the HMAC-SHA256 algorithm before storing them.	Account & Security Rules
BR-53	The customer can send a request to register as a restaurant owner; an Admin must approve this request.	Account & Security Rules
BR-54	The customer is only allowed to message others when they are friends.	Account & Security Rules
BR-55	The customer is only allowed to message a restaurant when they have followed it.	Account & Security Rules
BR-56	The system must send notifications to the customer based on predefined types.	Account & Security Rules
BR-57	The restaurant is only allowed to operate after their business license is registered and approved by an Admin.	Account & Security Rules
BR-58	The customer can listen to blog content using Text-to-Speech if the browser/device supports Vietnamese voice.	Account & Security Rules
BR-59	When creating or updating items related to photos/videos, the system must compress the images and upload them to AWS S3.	Account & Security Rules
BR-60	The restaurant is only allowed to close when there are no active Reservations and no Orders in processing status.	Account & Security Rules
BR-61	The system only allows deleting a menu when the menu no longer contains any dishes.	Inventory & Menu Rules
BR-62	The system only allows deleting a dish when that dish is not in any order related to that dish.	Inventory & Menu Rules

2. Guest Requirements

2.1 Actors

To effectively define the functional requirements of the DineGO system, it is essential to first identify and describe the primary actors who interact with the system. These actors represent the different types of users, each with distinct roles, permissions, and usage contexts. The table below outlines the main system actors and their corresponding responsibilities:

Table 13: Description of System Actors

#	Actor	Description
1	Guest	A general visitor who browses the platform without creating an account. May search for restaurants, menus, or view ratings without interaction.
2	Customer	A registered user who can book tables, place orders, leave reviews, and manage personal account information.
3	Admin	The system administrator who manages user accounts, verifies restaurant owners, monitors activities, and ensures system stability.
4	Restaurant Owner	A business account holder who registers their restaurant, manages menu, confirms bookings, views customer feedback, and updates availability.

2.2 Use Cases

a) Diagram(s)

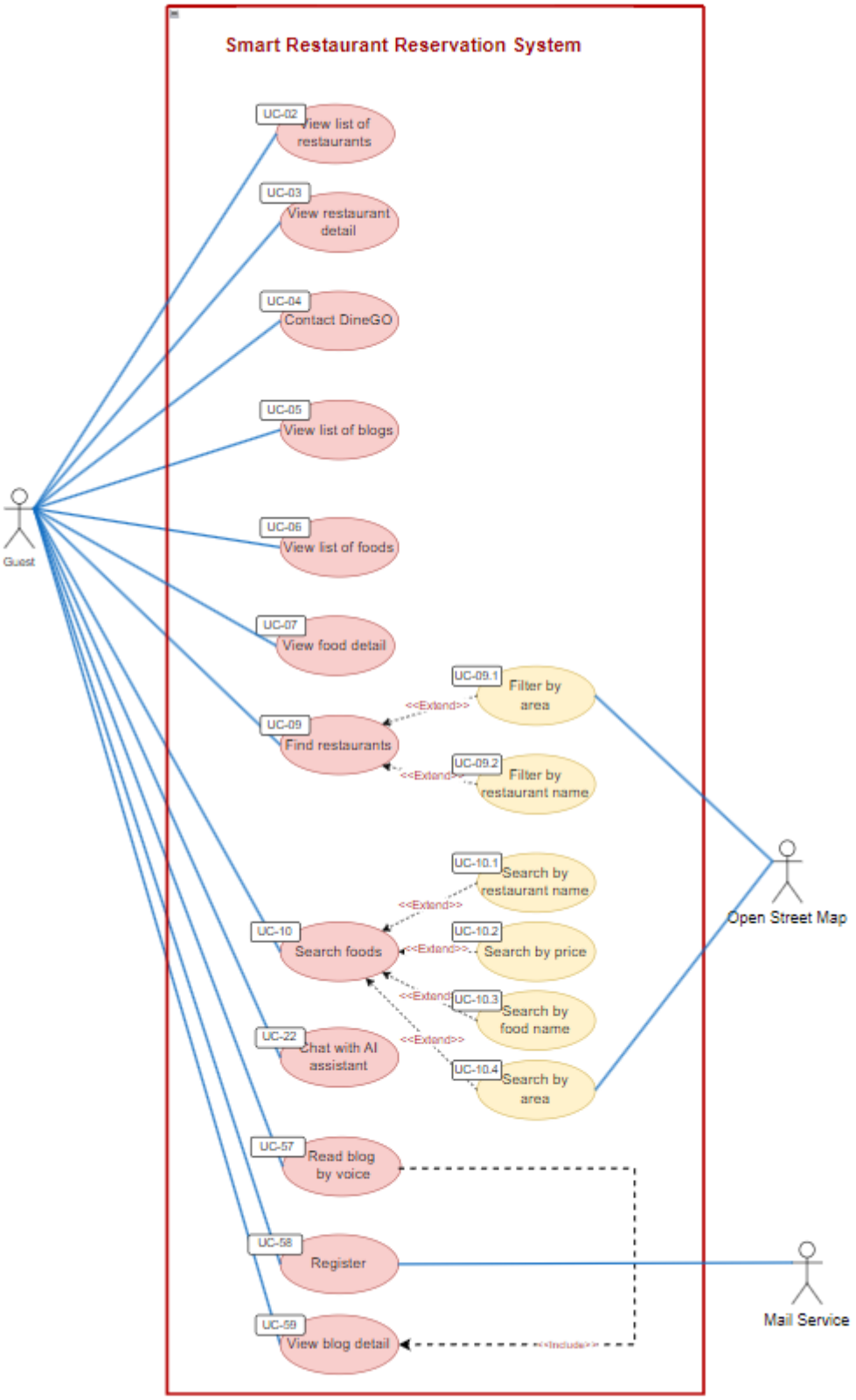


Figure 25: Use Case diagram of Guest.

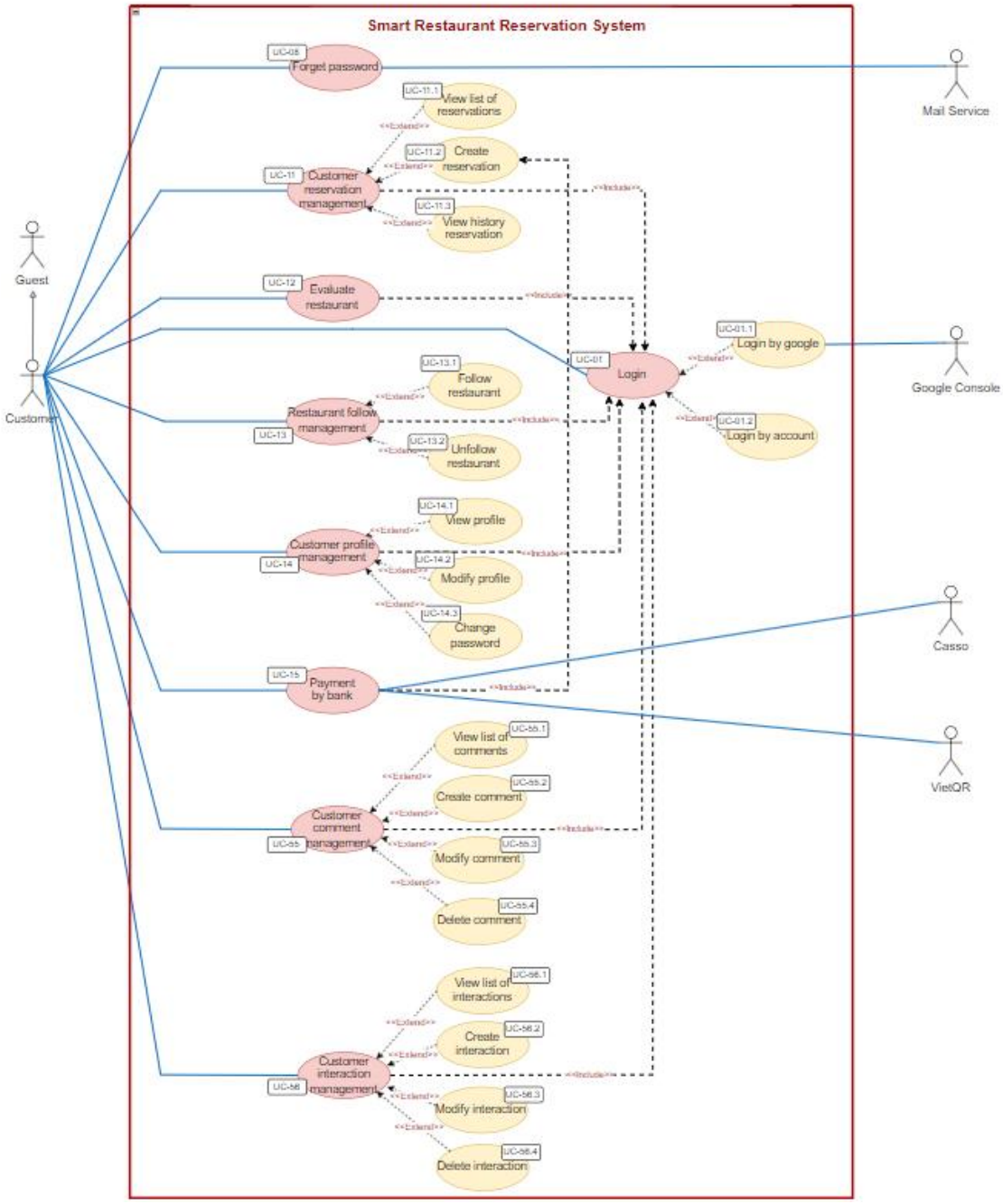


Figure 26: Use Case diagram of Customer. (1)

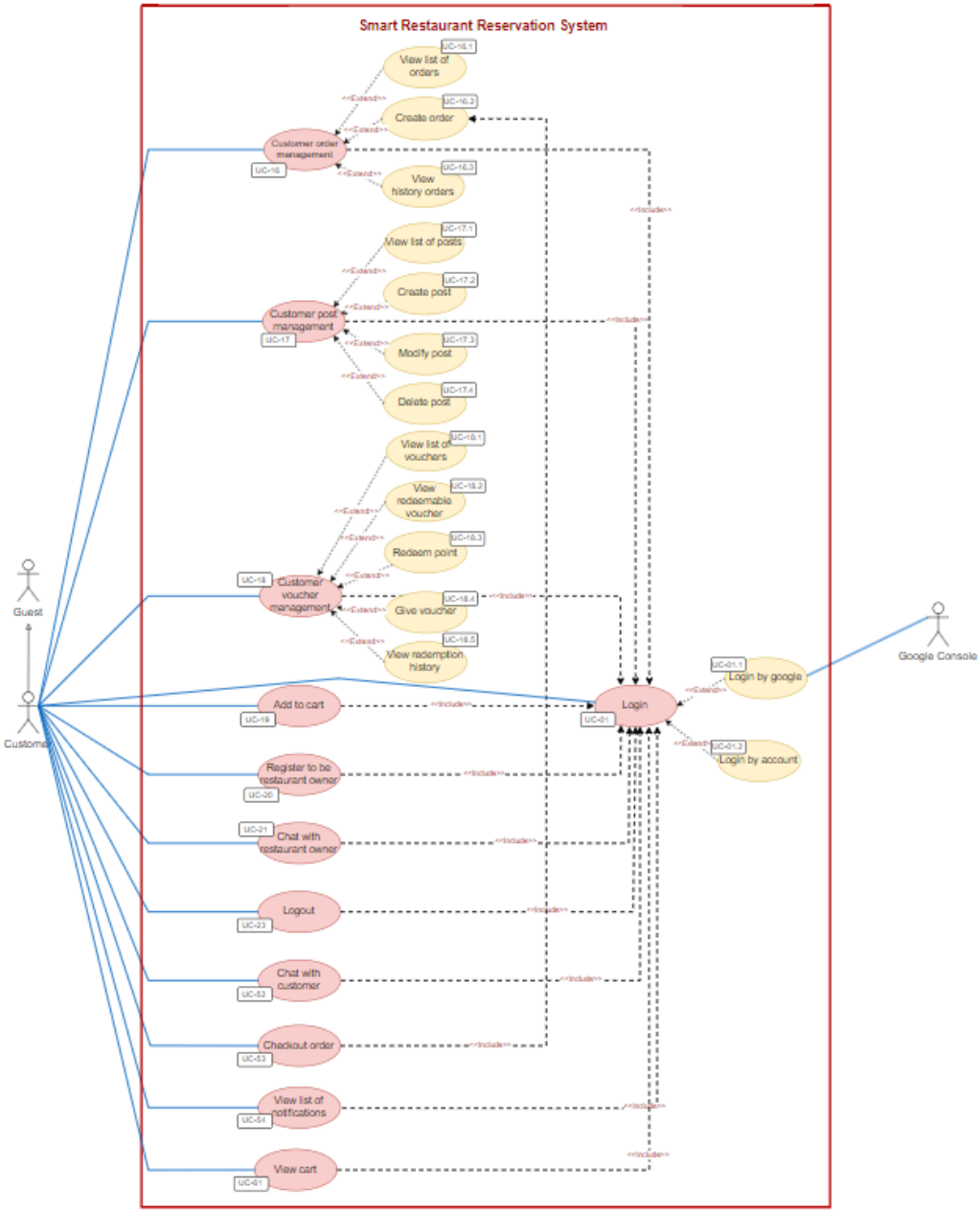


Figure 27: Use Case diagram of Customer. (2)

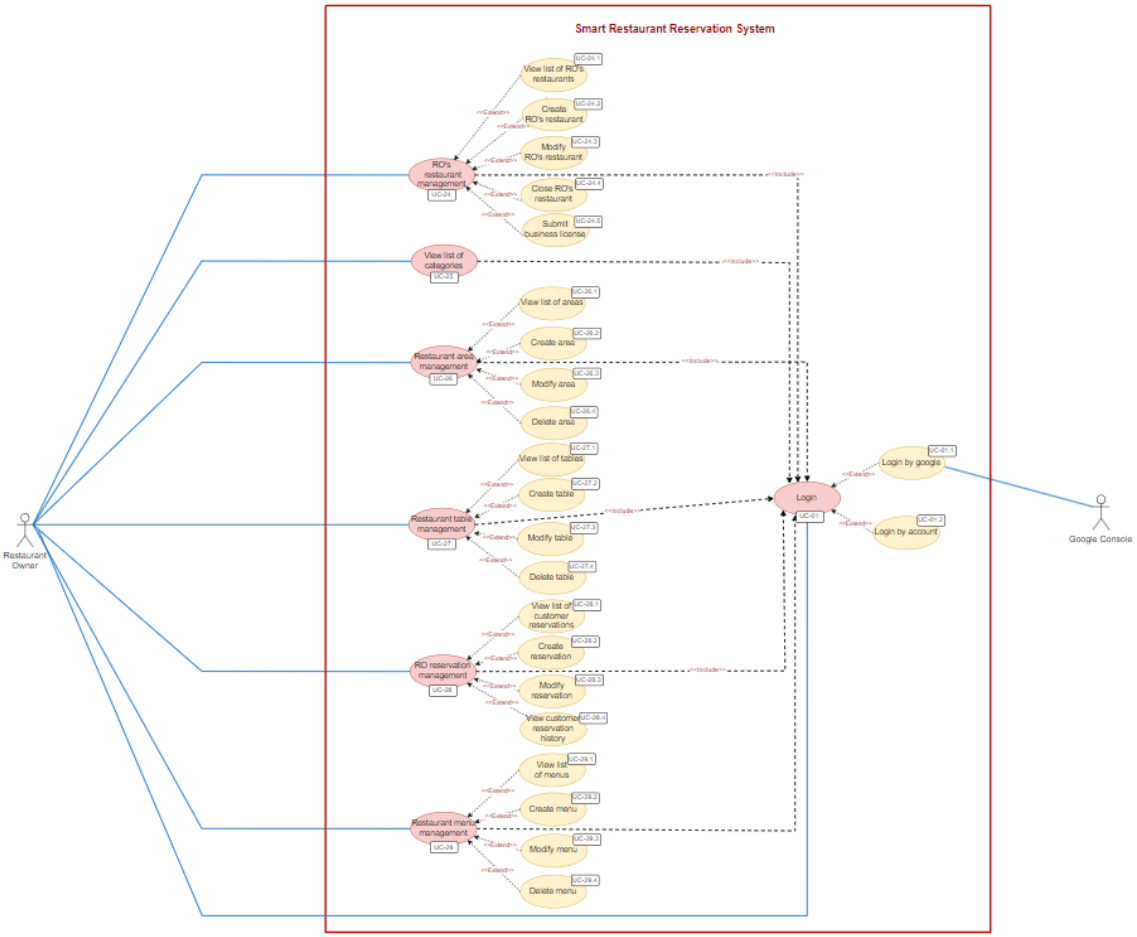


Figure 28*: Use Case diagram of Restaurant Owner. (1)*

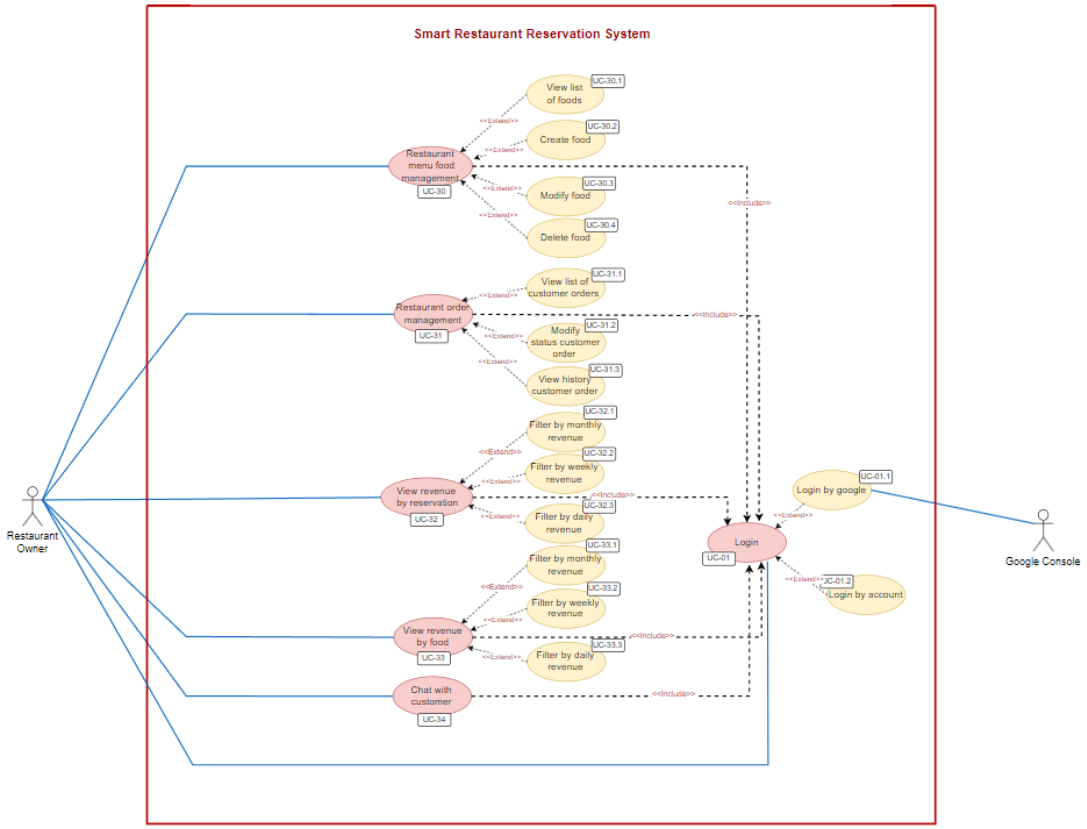


Figure 29: Use Case diagram of Restaurant Owner. (2)

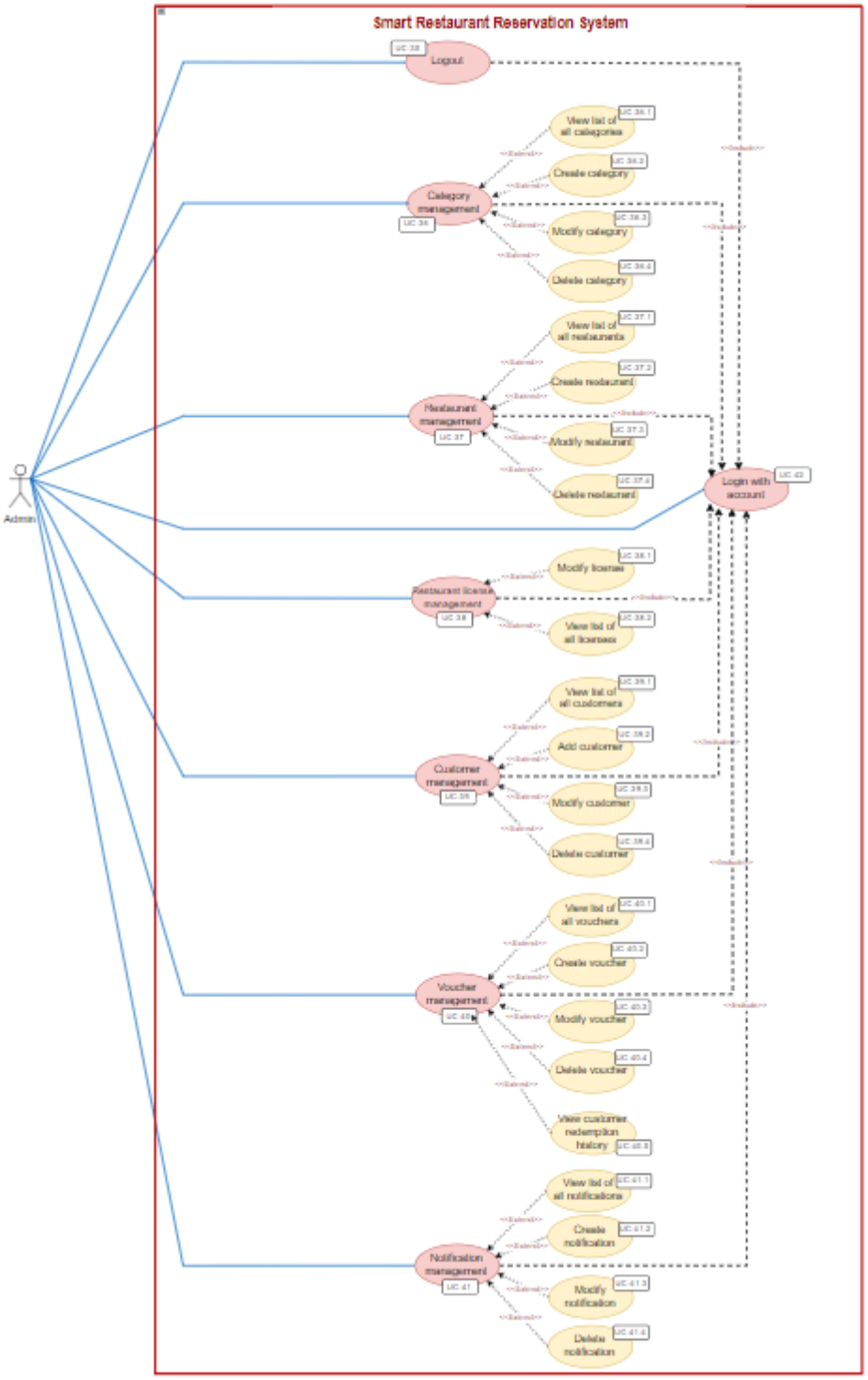


Figure 30: Use Case diagram of Admin. (1)

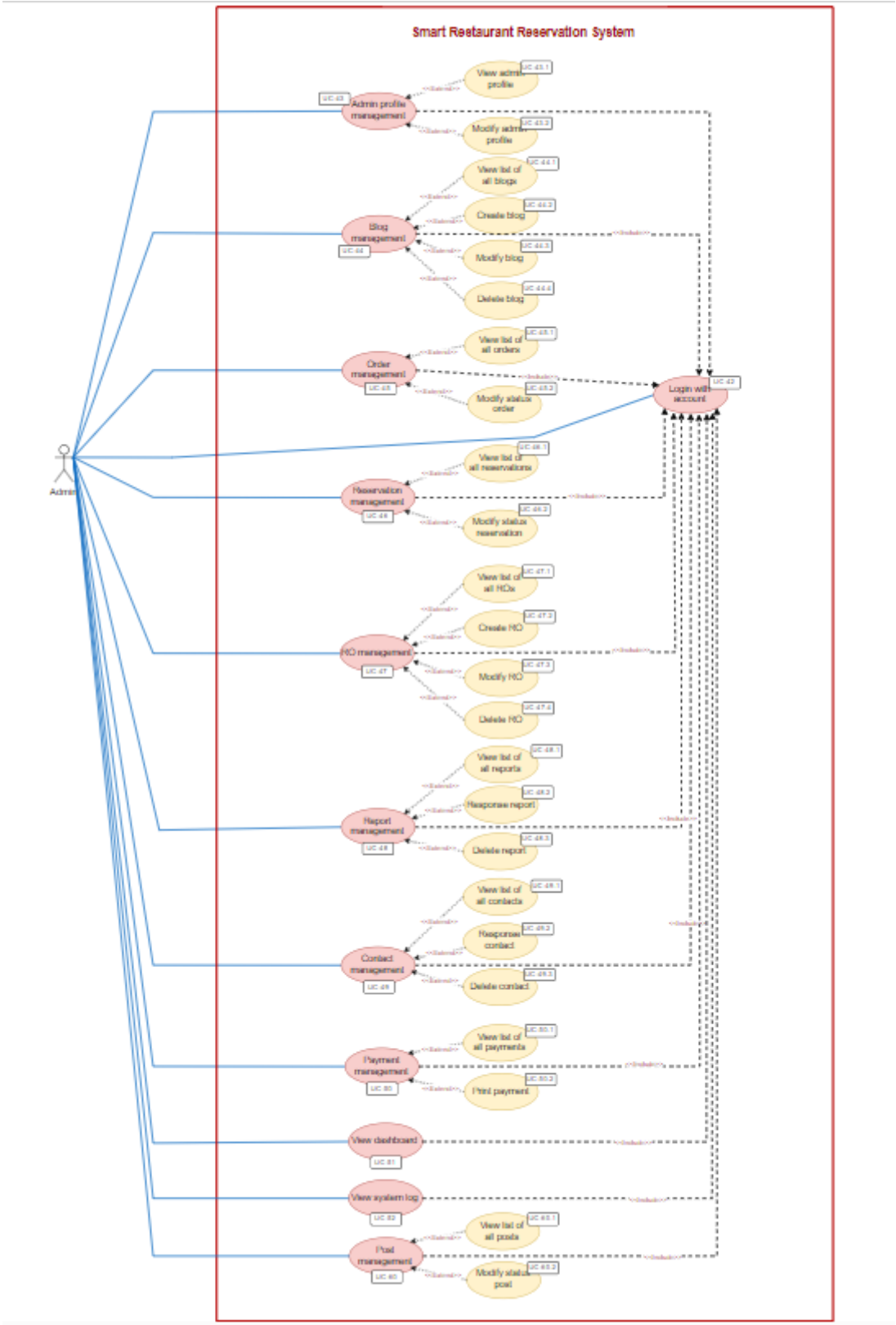


Figure 31: Use Case diagram of Admin. (2)

b) Descriptions

This section contains a detailed table that outlines the use case functions associated with the Guest role in the system. Each row represents a specific use case, including the Use Case ID, the title of the use case, the actor(s) involved, and a brief description of what the user is expected to do or what the system provides. These descriptions correlate directly with the diagrams presented earlier, offering a clearer understanding of how users interact with the system and the functionalities available to them.

Table 14: Use Case Descriptions for system

ID	Use Case	Actors	Use Case Description
UC-01	Login	Guest	The user logs into the system via Google or account credentials.
UC-01.1	Login by Google	Guest	Login using a Google account.
UC-01.2	Login by account	Guest	Login using email and password.
UC-02	View list of restaurants	Guest	View the list of available restaurants.
UC-03	View restaurant detail	Guest	View detailed information about a selected restaurant.
UC-04	Contact DineGO	Guest	Send inquiries, feedback, or requests to the DineGO system.
UC-05	View list of blogs	Guest	Read blog entries related to food or restaurants.
UC-06	View list of foods	Guest	View a list of food items from restaurants.
UC-07	View food detail	Guest	View full information about a selected dish.
UC-08	Forget password	Customer	Recover access to the system by resetting password.
UC-09	Find restaurant	Guest	Search for restaurants based on various criteria.
UC-09.1	Filter by area	Guest	Find restaurants in a specific geographic area.
UC-09.2	Filter by restaurant name	Guest	Find restaurants by entering their names.

ID	Use Case	Actors	Use Case Description
UC-10	Search food	Guest	Search for food items based on different criteria.
UC-10.1	Search by restaurant	Guest	Search food items by their associated restaurants.
UC-10.2	Search by price	Guest	Search food items by their price range.
UC-10.3	Search by food name	Guest	Search for a dish using its name.
UC-10.4	Search by area	Guest	Narrow down food search results by geographical region.
UC-11	Customer reservation management	Customer	Manage all reservation actions including create, modify, view, and cancel.
UC-11.1	View list of reservations	Customer	View current or past reservations.
UC-11.2	Create reservation	Customer	Create a new reservation for a restaurant.
UC-11.3	View history reservation	Customer	View past reservations.
UC-12	Evaluate restaurant	Customer	Provide feedback and ratings for restaurants.
UC-13	Restaurant follow management	Customer	Manage the list of restaurants the customer follows.
UC-13.1	Follow restaurant	Customer	Add a restaurant to the customer's follow list to receive updates.
UC-13.2	Unfollow restaurant	Customer	Remove a restaurant from the customer's follow list.
UC-14	Customer profile management	Customer	View and update user profile.
UC-14.1	View profile	Customer	Display personal information.
UC-14.2	Modify profile	Customer	Edit personal information.
UC-14.3	Change password	Customer	Change their customer password.
UC-15	Payment by bank	Customer	Pay for an order or reservation using bank transfer or online banking methods.

ID	Use Case	Actors	Use Case Description
UC-16	Customer order management	Customer	Manage all food orders including order, cancel, and view history.
UC-16.1	View list of orders	Customer	View current or past orders food.
UC-16.2	Create order	Customer	Place a new food order.
UC-16.3	View history order	Customer	View past orders.
UC-17	Customer post management	Customer	Create, edit, delete, or view posts.
UC-17.1	View list of posts	Customer	Read posted content.
UC-17.2	Create post	Customer	Write a new post.
UC-17.3	Modify post	Customer	Edit an existing post.
UC-17.4	Delete post	Customer	Remove an existing post.
UC-18	Customer voucher management	Customer	View, redeem, give, or manage vouchers.
UC-18.1	View list of vouchers	Customer	Customer views owned vouchers.
UC-18.2	View redeemable voucher	Customer	Customer views vouchers available for redemption with points.
UC-18.3	Redeem point	Customer	Customer exchanges points for a voucher.
UC-18.4	Give voucher	Customer	Customer gives a voucher to a friend.
UC-18.5	View redemption history	Customer	Customer views past point redemption activities.
UC-19	Add to cart	Customer	Add food items to shopping cart.
UC-20	Register to be restaurant owner	Customer	Apply to become a restaurant owner.
UC-21	Chat with restaurant owner	Customer	Send messages to restaurant owners.
UC-22	Chat with AI assistant	Guest	Interact with AI assistants for support.
UC-23	Logout	Customer	Log out of the system

ID	Use Case	Actors	Use Case Description
UC-24	RO's restaurant management	Restaurant Owner	Create, view, update, and delete restaurant information.
UC-24.1	View list of RO's restaurants	Restaurant Owner	Display a list of registered restaurants.
UC-24.2	Create RO's restaurant	Restaurant Owner	Add new restaurants to the system.
UC-24.3	Modify RO's restaurant	Restaurant Owner	Edit restaurant information.
UC-24.4	Close RO's restaurant	Restaurant Owner	Close or temporarily suspend the restaurant.
UC-24.5	Submit business license	Restaurant Owner	Upload or attach the business license for system/admin verification.
UC-25	View list of categories	Restaurant Owner	Display available food categories.
UC-26	Restaurant area management	Restaurant Owner	Manage different areas of the restaurant (e.g., Area A, Garden, Floor 1).
UC-26.1	View list of areas	Restaurant Owner	View the list of areas.
UC-26.2	Create area	Restaurant Owner	Create a new area.
UC-26.3	Modify area	Restaurant Owner	Edit the information of an existing area.
UC-26.4	Delete area	Restaurant Owner	Delete an area from the restaurant.
UC-27	Restaurant table management	Restaurant Owner	Manage tables within each restaurant area.
UC-27.1	View list of tables	Restaurant Owner	Display table list.
UC-27.2	Create table	Restaurant Owner	Add a new table.
UC-27.3	Modify table	Restaurant Owner	Edit table information.
UC-27.4	Delete table	Restaurant Owner	Delete a table.
UC-28	RO reservation management	Restaurant Owner	Manage reservation records.

ID	Use Case	Actors	Use Case Description
UC-28.1	View list of customer reservations	Restaurant Owner	Display reservation list.
UC-28.2	Create reservation	Restaurant Owner	Add a reservation manually.
UC-28.3	Modify reservation	Restaurant Owner	Edit reservation detail.
UC-28.4	View customer reservation history	Restaurant Owner	View past reservations.
UC-29	Restaurant menu management	Restaurant Owner	Manage menus of the restaurant (create, edit, delete).
UC-29.1	View list of menus	Restaurant Owner	View the list of existing menus.
UC-29.2	Create menu	Restaurant Owner	Create a new menu with name, type, and cover image.
UC-29.3	Modify menu	Restaurant Owner	Edit the detail of an existing menu.
UC-29.4	Delete menu	Restaurant Owner	Delete a menu from the restaurant.
UC-30	Restaurant menu food management	Restaurant Owner	Create, edit, or delete food items.
UC-30.1	View list of foods	Restaurant Owner	Display food items.
UC-30.2	Create food	Restaurant Owner	Add a new food item.
UC-30.3	Modify food	Restaurant Owner	Edit food item detail.
UC-30.4	Delete food	Restaurant Owner	Remove a food item.
UC-31	Restaurant order management	Restaurant Owner	Manage food orders.
UC-31.1	View list of customer order	Restaurant Owner	See list of food orders.
UC-31.2	Modify status customer order	Restaurant Owner	Update the status of a food order (e.g., pending, in progress, delivered).
UC-31.3	View history customer orders	Restaurant Owner	View the history of completed and cancelled food orders.
UC-32	View revenue by reservation	Restaurant Owner	Check revenue based on reservations.

ID	Use Case	Actors	Use Case Description
UC-32.1	Filter by monthly revenue	Restaurant Owner	View monthly earnings from reservations.
UC-32.2	Filter by weekly revenue	Restaurant Owner	View weekly earnings from reservations.
UC-32.3	Filter by daily revenue	Restaurant Owner	View daily earnings from reservations.
UC-33	View revenue by food	Restaurant Owner	Check revenue based on food items.
UC-33.1	Filter by monthly revenue	Restaurant Owner	View monthly earnings from food items.
UC-33.2	Filter by weekly revenue	Restaurant Owner	View weekly earnings from food items.
UC-33.3	Filter by daily revenue	Restaurant Owner	View daily earnings from food items.
UC-34	Chat with customer	Restaurant Owner	Communicate with customers via chat.
UC-35	Logout	Admin	Log out of the system and end the current admin session.
UC-36	Category management	Admin	Manage restaurant categories in the system: view list, create, modify, delete.
UC-36.1	View list of all categories	Admin	Display all restaurant categories.
UC-36.2	Create category	Admin	Add a new category for restaurants.
UC-36.3	Modify category	Admin	Edit the detail of an existing category.
UC-36.4	Delete category	Admin	Remove a category from the system.
UC-37	Restaurant management	Admin	Manage restaurant approvals and blocks.
UC-37.1	View list of all restaurants	Admin	View all registered restaurants.

ID	Use Case	Actors	Use Case Description
UC-37.2	Create restaurant	Admin	Add a new restaurant to the system.
UC-37.3	Modify restaurant	Admin	Edit restaurant detail.
UC-37.4	Delete restaurant	Admin	Remove a restaurant from the system.
UC-38	Restaurant license management	Admin	Handle license approval or rejection.
UC-38.1	Modify license	Admin	Approve/Reject restaurant license applications.
UC-38.2	View list of all licenses	Admin	Display all submitted business licenses.
UC-39	Customer management	Admin	View, block or delete customers.
UC-39.1	View list of all customers	Admin	See all customer accounts.
UC-39.2	Add customers	Admin	Create a new customer account.
UC-39.3	Modify customer	Admin	Edit customer information.
UC-39.4	Delete customer	Admin	Remove a customer account from the system.
UC-40	Voucher management	Admin	Create or delete discount vouchers.
UC-40.1	View list of all vouchers	Admin	View all available vouchers.
UC-40.2	Create voucher	Admin	Add a new voucher.
UC-40.3	Modify voucher	Admin	Edit detail of an existing voucher.
UC-40.4	Delete voucher	Admin	Remove a voucher.
UC-40.5	View customer redemption history	Admin	View all history of customer redemption.
UC-41	Notification management	Admin	Handle creation or deletion of notifications.
UC-41.1	View list of all notifications	Admin	View all notifications sent.
UC-41.2	Create notification	Admin	Compose and send a new notification.

ID	Use Case	Actors	Use Case Description
UC-41.3	Modify notification	Admin	Edit detail of an existing notification.
UC-41.4	Delete notification	Admin	Remove a notification.
UC-42	Login with account	Admin	Admin logs into the system using credentials.
UC-43	Admin profile management	Admin	Manage admin profile detail.
UC-43.1	View admin profile	Admin	Display admin profile information.
UC-43.2	Modify admin profile	Admin	Edit admin profile detail.
UC-44	Blog management	Admin	Manage blog posts in the system.
UC-44.1	View list of all blogs	Admin	Display a list of all blogs.
UC-44.2	Create blog	Admin	Add a new blog post.
UC-44.3	Modify blog	Admin	Edit an existing blog post.
UC-44.4	Delete blog	Admin	Remove a blog post from the system.
UC-45	Order management	Admin	Manage all customer orders.
UC-45.1	View list of all orders	Admin	Display all orders in the system.

ID	Use Case	Actors	Use Case Description
UC-45.2	Modify status orders	Admin	Update the order status.
UC-46	Reservation management	Admin	Manage all reservations in the system.
UC-46.1	View list of all reservations	Admin	Display all reservations.
UC-46.2	Modify status reservation	Admin	Update the reservation status.
UC-47	RO management	Admin	Manage restaurant owner accounts.
UC-47.1	View list of all ROs	Admin	Display all restaurant owners.
UC-47.2	Create RO	Admin	Add a new restaurant owner account.
UC-47.3	Modify RO	Admin	Edit restaurant owner detail.
UC-47.4	Delete RO	Admin	Remove a restaurant owner account.
UC-48	Report management	Admin	Manage reports submitted by customers or restaurant owners.
UC-48.1	View list of all reports	Admin	Display all reports in the system.
UC-48.2	Response report	Admin	Respond to a submitted report.
UC-48.3	Delete report	Admin	Remove a report from the system.

ID	Use Case	Actors	Use Case Description
UC-49	Contact management	Admin	Manage contact messages submitted via the system.
UC-49.1	View list of all contacts	Admin	Display all contact messages.
UC-49.2	Response contact	Admin	Respond to a contact message.
UC-49.3	Delete contact	Admin	Remove a contact message from the system.
UC-50	Payment management	Admin	Manage payment transactions.
UC-50.1	View list of all payments	Admin	Display all payment transactions.
UC-50.2	Print payment	Admin	Print payment detail.
UC-51	View dashboard	Admin	Display the admin dashboard with key metrics and statistics.
UC-52	View system log	Admin	Admin views the history of system activities (e.g., account actions, modifications).
UC-53	Checkout order	Customer	Complete payment and confirm an order for delivery or pickup.
UC-54	View list of notifications	Customer	View system notifications such as promotions, order updates, or reminders.
UC-55	Customer comment management	Customer	Manage all comments made by customers in the system.
UC-55.1	View list of comments	Customer	Display all customer comments.

ID	Use Case	Actors	Use Case Description
UC-55.2	Create comment	Customer	Add a new comment on behalf of a customer (or for testing purposes).
UC-55.3	Modify comment	Customer	Edit the content of a customer's comment.
UC-55.4	Delete comment	Customer	Remove a comment from the system.
UC-56	Customer interaction management	Customer	Manage all likes given by customers.
UC-56.1	View list of interactions	Customer	Display all customer likes.
UC-56.2	Create interaction	Customer	Add a like on behalf of a customer (or for testing purposes).
UC-56.3	Modify interaction	Customer	Change the status or type of a like.
UC-56.4	Delete interaction	Customer	Remove a like from the system.
UC-57	Read blog by voice	Guest	Guest listens to blog content via text-to-speech.
UC-58	Register	Guest	Guest creates a new customer account
UC-59	View blog detail	Guest	Customer opens and reads a specific blog in detail.
UC-60	Post management	Admin	Manage all posts, including viewing and updating their status.
UC-60.1	View list of all posts	Admin	View the full list of posts with detail such as author, content, time, and status.

ID	Use Case	Actors	Use Case Description
UC-60.2	Modify status post	Admin	Change a post's status (approved, rejected) and notify the author.
UC-61	View cart	Customer	Display cart of customer

2.3 Use Cases Detail

2.3.1 Login as customer

UC ID and Name:	**UC-01 Login**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest	Secondary Actors:	Google Console
Trigger:	The user clicks the "Đăng nhập" link on the navigation bar		
Description:	A function for users to login and access to web pages.		
Preconditions:	The user must have a valid account.		
Postconditions:	The user is redirected to the main page with appropriate permissions.		
Normal flow:	**1.0 Login by customer account.** 1. Guest accesses the login page. 2. Guest selects the login method. 2.1 Guests choose "Login by Customer Account". 3. Guest enters username and password. 4. Guest clicks the "Login" button. 5. The system validates the credentials. 6. The system redirects the user to the home page.		
Alternative Flows:	**2.2 Login by Google account** 1. At step 2.2, the user clicks the "Login by Google" button. 2. The system redirects the user to Google OAuth authentication page. 3. The user authenticates using their Google account. 4. Google sends the authentication token back to the system. 5. The system validates the token.		

	6. The system redirects the user to the home page.
Exceptions:	**1.0.E1 Guest enters username or password fields blank.** Inform users cannot leave the input field blank. **1.0.E2 Guest input the wrong username or password.** Inform users have input an invalid username or password.
Priority:	High
Frequency of Use:	Regularly
Business Rules:	N/A
Other Information:	Endpoint API: /api/customer/login Passwords need to be encrypted before being saved to the database.
Assumptions:	The account must exist and not be blocked by the admin.

2.3.2 View list of restaurants

UC ID and Name:	**UC-02 View list of restaurants**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	None
Trigger:	The user clicks the “Nhà hàng gần bạn” link on the navigation bar.		
Description:	This use case allows users to view a list of available restaurants in the system.		
Preconditions:	The system is online and accessible.		
Postconditions:	The user is presented with a list of restaurants, including basic info (name, address, rating, etc….).		
Normal flow:	**2.0 View restaurant list** 1. Guest clicks on “Nhà hàng”. 2. The system fetches and displays a list of restaurants.		
Alternative Flows:	N/A		
Exceptions:	**2.0.E1 System error or API failure**		

UC ID and Name:	**UC-02 View list of restaurants**
	Inform “Unable to load data. Please try again later.”
Priority:	Medium
Frequency of Use:	Regularly
Business Rules:	N/A
Other Information:	Connected to restaurant API: /api/restaurant
Assumptions:	The user has an internet connection and the system backend is responsive.

2.3.3 View restaurant detail

UC ID and Name:	**UC-03 View restaurant detail**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	None
Trigger:	The user clicks on a specific restaurant in the restaurant list.		
Description:	This use case allows users to view detailed information about a selected restaurant.		
Preconditions:	The user has accessed the restaurant list.		
Postconditions:	Detailed information about the restaurant is displayed, including description, location, operating hours, price, evaluation of customer, etc….		
Normal flow:	**3.0 View restaurant detail** 1. The user selects a restaurant from the list. 2. System retrieves restaurant data. 3. System displays full detail of the selected restaurant.		
Alternative Flows:	None		
Exceptions:	**3.0.E1 System error or API failure** Inform “Unable to load data. Please try again later.”		
Priority:	Medium		

UC ID and Name:	**UC-03 View restaurant detail**
Frequency of Use:	Regularly
Business Rules:	N/A
Other Information:	Uses endpoint /api/restaurant/{id} to fetch data.
Assumptions:	Restaurant ID is valid and exists in the system.

2.3.4 Contact DineGO

UC ID and Name:	**UC-04 Contact DineGO**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	None
Trigger:	The user clicks “Liên hệ” link on the navigation bar.		
Description:	A function for users to send feedback, questions, or support requests via the contact form.		
Preconditions:	The user must fill in all required fields in the contact form.		
Postconditions:	The system sends the contact request and displays a confirmation message to the user.		
Normal flow:	**4.0 Send a contact request.** 1. User accesses the Contact page. 2. User enters their full name. 3. User enters their email. 4. User enters the contact title. 5. User enters the contact content. 6. User clicks the “Gửi liên hệ” button. 7. The system saves the request and shows a confirmation message.		

UC ID and Name:	**UC-04 Contact DineGO**
Alternative Flows:	N/A
Exceptions:	**4.0.E1 Guest leaves required fields blank.** Inform the user that all required fields must be filled. **4.0.E2 Invalid email format.** Inform the user that the email format is invalid.
Priority:	Medium
Frequency of Use:	Sometimes
Business Rules:	N/A
Other Information:	N/A
Assumptions:	The system stores the contact request in the database for admin review.

2.3.5 View list of blogs

UC ID and Name:	**UC-05 View list of blogs**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	None
Trigger:	Guest clicks on “Tin tức” link on the side navigation bar.		
Description:	This use case allows users to view available blogs in the system.		
Preconditions:	The system is online and accessible.		
Postconditions:	The user is presented with blogs, including info (title, name of blogger, date posted, etc….).		
Normal flow:	**5.0 View blogs** 1. Guest clicks on “Tin tức”. 2. System fetches and displays blogs.		
Alternative Flows:	N/A		

UC ID and Name:	**UC-05 View list of blogs**
Exceptions:	**5.0.E1 System error or API failure** Inform “Unable to load data. Please try again later.”
Priority:	Medium
Frequency of Use:	Regularly
Business Rules:	N/A
Other Information:	Connected to restaurant API: /api/blog
Assumptions:	Guest has an internet connection and the system backend is responsive.

2.3.6 View list of foods

UC ID and Name:	**UC-06 View list of foods**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	None
Trigger:	Guest clicks on “Món ăn” link on the side navigation bar.		
Description:	This use case allows users to view a list of available foods in the system.		
Preconditions:	The system is online and accessible.		
Postconditions:	The user is presented with a list of foods, including basic info (name foods, name restaurant, address, rating, price, etc….).		
Normal flow:	**6.0 View foods list** 1. Guest clicks on “Món ăn”. 2. System fetches and displays a list of foods.		
Alternative Flows:	N/A		
Exceptions:	**6.0.E1 System error or API failure** Inform “Unable to load data. Please try again later.”		
Priority:	Medium		

UC ID and Name:	**UC-06 View list of foods**
Frequency of Use:	Regularly
Business Rules:	N/A
Other Information:	Connected to restaurant API: /api/food
Assumptions:	Guest has an internet connection and the system backend is responsive.

2.3.7 View food detail

UC ID and Name:	**UC-07 View food detail**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	None
Trigger:	Guest clicks on a specific food in the food list.		
Description:	This use case allows users to view detailed information about a selected food.		
Preconditions:	Guest has accessed the foods list.		
Postconditions:	Detailed information about the food is displayed, including description, location, price, evaluation of customer, etc….		
Normal flow:	**7.0 View food detail** 1. Guest selects a food from the list. 2. System retrieves food data. 3. System displays full detail of the selected food.		
Alternative Flows:	N/A		
Exceptions:	**7.0.E1 System error or API failure** Inform “Unable to load data. Please try again later.”		
Priority:	Medium		
Frequency of Use:	Regularly		
Business Rules:	BR-45		

UC ID and Name:	**UC-07 View food detail**
Other Information:	Uses endpoint: /api/food/{id} to fetch data.
Assumptions:	Food ID is valid and exists in the system.

2.3.8 Forget password

UC ID and Name:	**UC-08 Forget password**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest	Secondary Actors:	Mail Service
Trigger:	Guest clicks on the "Quên mật khẩu" link on the login page.		
Description:	This use case allows users to reset their password when they forget it. The system verifies the user's email and sends a newly generated random password via the Mail Service to the registered email.		
Preconditions:	The user has a registered email in the system. Mail Service is available.		
Postconditions:	If the email is valid and Mail Service delivers successfully → The user receives a new random password via email and can log in using it. If the email is invalid or not found → The system informs the user and does not send anything.		
Normal flow:	**8.0 Reset password via email** 1. Guest clicks "Quên mật khẩu?" link. 2. Guest enters a registered email. 3. Guest clicks "Gửi yêu cầu" button. 4. The system verifies the email. 5. The system generates a new random password. 6. The system sends the new password to the user's email via Mail Service. 7. The user logs in using the new password.		
Alternative Flows:	N/A		
Exceptions:	**8.0.E1 Email not registered** → System shows message: "Địa chỉ email không hợp lệ." **8.0.E2 Mail Service unavailable:** → System shows message: **"**Không thể gửi email. Vui lòng thử lại sau.**"**		
Priority:	High		

UC ID and Name:	**UC-08 Forget password**
Frequency of Use:	Sometimes
Business Rules:	BR-50, BR-52
Other Information:	Endpoint: /api/auth/forgot-password
Assumptions:	Guests remember their registered email and have access to their email.

2.3.9 Filter by area in Restaurant page

UC ID and Name:	**UC-09.1 Filter by area**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	Open Street Map
Trigger:	The user clicks “Chọn vị trí của bạn” button and selects a region on the map.		
Description:	A function that allows users to filter and display restaurants based on the selected region.		
Preconditions:	The system must have restaurant data with associated region/location information.		
Postconditions:	The system displays the list of restaurants corresponding to the selected region.		
Normal flow:	**9.1 Filter restaurants by area.** 1. Guest clicks the “Chọn vị trí của bạn” button. 2. The system displays a map for location selection. 3. Guest selects a location or clicks “Chọn vị trí hiện tại”. 4. Guest clicks the “Xác nhận” button. 5. The system filters and displays restaurants in the selected region.		
Alternative Flows:	**9.1 Search by region name in text box.** 1. Guest types a region name in the location input box. 2. The system filters and displays restaurants in that region without using the map.		
Exceptions:	**9.0.E1 No restaurants found in the selected region.** Inform the user that there are no available restaurants in this region. **9.0.E2 Location selection canceled.** Return to the unfiltered restaurant list.		

UC ID and Name:	**UC-09.1 Filter by area**
Priority:	High
Frequency of Use:	Regularly
Business Rules:	BR-46, BR-47
Other Information:	API Endpoint: /api/restaurant/search?region={regionName} (GET) – Returns a list of restaurants filtered by region.
Assumptions:	The system uses GPS or map coordinates to determine and filter restaurant locations accurately.

2.3.10 Filter by restaurant name in Restaurant page

UC ID and Name:	**UC-09.2 Filter by restaurant name**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	None
Trigger:	Guest enters a restaurant name (or part of it) into the search input on the Restaurant page.		
Description:	This use case allows users to search restaurants by entering keywords in their names.		
Preconditions:	Restaurant page is visible and connected to the search API.		
Postconditions:	A filtered list of restaurants matching the search keyword is shown.		
Normal flow:	**9.2 Search by restaurant name** 1. Guest enters the name of the restaurant into search input. 2. System sends requests to the search backend. 3. System returns and displays matching restaurants.		
Alternative Flows:	N/A		
Exceptions:	**9.2.E1** No matching restaurants found Inform “Không tìm thấy nhà hàng.”		
Priority:	High		

UC ID and Name:	**UC-09.2 Filter by restaurant name**
Frequency of Use:	Regularly
Business Rules:	N/A
Other Information:	Endpoint: /api/restaurant/search?name=...
Assumptions:	Guest knows at least part of the restaurant name.

2.3.11 Search by restaurant name in Food page

UC ID and Name:	**UC-10.1 Search by restaurant name**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	None
Trigger:	The user enters a restaurant name in the advanced search form.		
Description:	Allows users to filter foods by the name of the restaurant providing them.		
Preconditions:	The system must have restaurant and food data linked together.		
Postconditions:	The system displays a list of foods belonging to the specified restaurant.		
Normal flow:	**10.1 Search by restaurant name** 1. Guest opens advanced search. 2. Guest enters the restaurant name. 3. Guest clicks the “Lọc” button. 4. The system displays foods from the entered restaurant.		
Alternative Flows:	N/A		
Exceptions:	**10.1.E1 No matching restaurant found.** Inform the user that no foods match the entered restaurant name.		

UC ID and Name:	**UC-10.1 Search by restaurant name**
Priority:	High
Frequency of Use:	Regularly
Business Rules:	BR-45
Other Information:	Endpoint: /api/food/search?res_name=...
Assumptions:	The restaurant name must be stored in a consistent format for searching.

2.3.12 Search by price in Food page

UC ID and Name:	**UC-10.2 Search by price**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	None
Trigger:	The user enters a minimum and/or maximum price in the advanced search form.		
Description:	Allows users to filter foods based on a specific price range.		
Preconditions:	The system must store price information for each food item.		
Postconditions:	The system displays a list of foods within the specified price range.		
Normal flow:	**10.2 Search by price** 1. Guest opens advanced search. 2. Guest enters a minimum and/or maximum price. 3. Guest clicks the “Lọc” button. 4. The system displays foods within the entered price range.		

UC ID and Name:	**UC-10.2 Search by price**
Alternative Flows:	N/A
Exceptions:	**10.2.E1 No foods found in the specified range.** Inform the user that no foods match the filter criteria.
Priority:	High
Frequency of Use:	Regularly
Business Rules:	BR-45
Other Information:	API Endpoint: /api/food/search?minPrice={minPrice}&maxPrice={maxPrice}
Assumptions:	Prices are stored in the same currency for all items.

2.3.13 Search by food name in Food page

UC ID and Name:	**UC-10.3 Search by food name**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	None
Trigger:	The user enters a food name in the advanced search form.		
Description:	Allows users to filter and search for foods by their name.		
Preconditions:	The system must store names for all food items.		
Postconditions:	The system displays a list of foods matching the entered name.		
Normal flow:	**10.3 Search by food name** 1. Guest opens advanced search. 2. Guest enters the food name.		

UC ID and Name:	**UC-10.3 Search by food name**
	3. Guest clicks the “Lọc” button. 4. The system displays foods matching the entered name.
Alternative Flows:	N/A
Exceptions:	**10.3.E1 No foods found with the entered name.** Inform the user that no foods match the entered name.
Priority:	High
Frequency of Use:	Regularly
Business Rules:	BR-45
Other Information:	API Endpoint: /api/food/search?foodName={foodName}
Assumptions:	Food names are stored in a consistent format for searching.

2.3.14 Search by area in Food page

UC ID and Name:	**UC-10.4 Search by area**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest, Customer	Secondary Actors:	Open Street Map
Trigger:	The user clicks "Chọn vị trí của bạn" button and selects a area on the map.		
Description:	A function that allows users to filter and display restaurants based on the selected area.		
Preconditions:	The system must have restaurant data with associated area/location information.		
Postconditions:	The system displays the list of restaurants corresponding to the selected area.		

UC ID and Name:	**UC-10.4 Search by area**
Normal flow:	**10.4 Search by area.** 1. Guest clicks the "Chọn vị trí của bạn" button. 2. The system displays a map for location selection. 3. Guest selects a location or clicks "Chọn vị trí hiện tại". 4. Guest clicks the "Xác nhận" button. 5. The system filters and displays restaurants in the selected region.
Alternative Flows:	**10.4.1 Search by area name in text box.** 1. Guest types a area name in the location input box. 2. The system filters and displays restaurants in that region without using the map.
Exceptions:	**10.4.E1 No restaurants found in the selected area.** Inform the user that there are no available restaurants in this region. **10.4.E2 Location selection canceled.** Return to the unfiltered restaurant list.
Priority:	High
Frequency of Use:	Regularly
Business Rules:	BR-45, BR-46, BR-47
Other Information:	API Endpoint: /api/restaurant/search?area={areaName}
Assumptions:	The system uses GPS or map coordinates to determine and filter restaurant locations accurately.

2.3.15 View list of reservations

UC ID and Name:	**UC-11.1 View list of reservations**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The Customer clicks the “Theo dõi đơn đặt bàn” menu item in the profile dropdown.		
Description:	This use case allows users to view their current reservations.		

UC ID and Name:	**UC-11.1 View list of reservations**
Preconditions:	The user is logged in and has existing reservations.
Postconditions:	A list of active reservations is displayed to the user.
Normal flow:	**11.1 View reservation** 1. Customer accesses the reservation section in the profile page . 2. System retrieves reservations. 3. System displays them in a list.
Alternative Flows:	N/A
Exceptions:	**11.1.E1 No active reservations** Inform: “Bạn chưa có đặt chỗ nào.”
Priority:	High
Frequency of Use:	Regularly
Business Rules:	BR-09, BR-10, BR-12, BR-14
Other Information:	Endpoint: /api/reservation/active?customerId=...
Assumptions:	Guest is authenticated and has permission to view own data.

2.3.16 Create reservation

UC ID and Name:	**UC-11.2 Create reservation**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The customer selects a restaurant and clicks the "Đặt chỗ ngay" button.		

UC ID and Name:	**UC-11.2 Create reservation**
Description:	Allows customers to create a reservation for a selected restaurant, including choosing date, time, area, and table.
Preconditions:	The customer must be logged in and the restaurant must have available tables.
Postconditions:	A new reservation record is created in the system.
Normal flow:	**11.2 Create reservation** 1. Customer accesses the reservation page of the selected restaurant. 2. Customer selects arrival date and time. 3. Customer selects an area within the restaurant. 4. The system displays available tables in that area. 5. Customer selects a table. 6. Customer optionally enters notes. 7. Customer clicks the “Thanh toán bằng QR” button. 8. The system creates a reservation and processes payment if applicable. 9. The system confirms the reservation and displays the detail.
Alternative Flows:	**11.2.1 Reservation without payment.** 1. Customer follows steps 1–7. 2. The system creates a reservation with status "Pending Payment".
Exceptions:	**11.2.E1 No tables available.** Inform the customer that no tables are available for the selected time and area. **11.2.E2 Invalid date/time.** Inform the customer to select a valid arrival date and time.
Priority:	High
Frequency of Use:	Regularly
Business Rules:	BR-02, BR-03, BR-04. BR-05, BR-06, BR-07, BR-11, BR-16
Other Information:	Endpoint: /api/reservation/create
Assumptions:	The system updates table availability in real-time to prevent double booking.

2.3.17 View history reservation

UC ID and Name:	**UC-11.3 View history reservation**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Guest clicks “Lịch sử đặt bàn” in their profile or menu		
Description:	Displays all past (served or canceled) reservations to the user.		
Preconditions:	Guest is logged in.		
Postconditions:	Reservation is marked as canceled.		
Normal flow:	**11.3 View history** 1. Guest accesses the reservation history. 2. System fetches all past records. 3. System shows them in a list view with filters.		
Alternative Flows:	N/A		
Exceptions:	**11.3.E1 No past reservations found** Inform: “Không có lịch sử đặt bàn.”		
Priority:	Low		
Frequency of Use:	Sometimes		
Business Rules:	BR-09		
Other Information:	Endpoint: /api/reservation/history?customerId=...		
Assumptions:	Guest has made at least one reservation before.		

2.3.18 Evaluate Restaurants

UC ID and Name:	**UC-12 Evaluate Restaurant**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025

UC ID and Name:	**UC-12 Evaluate Restaurant**		
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The customer navigates to a restaurant detail page and clicks on the rating section.		
Description:	Allows customers to rate and leave feedback for a restaurant after completing an order there.		
Preconditions:	The customer must be logged in and have completed at least one order at the restaurant.		
Postconditions:	The system saves the rating and review content to the database and updates the restaurant's average rating.		
Normal flow:	**12.0 Evaluate restaurant.** 1. Customer visits the restaurant detail page. 2. Customer navigates to the rating section. 3. Customer selects the number of stars. 4. Customer enters review content. 5. Customer clicks the "Gửi đánh giá" button. 6. The system saves the review and updates the restaurant's rating. 7. The review is displayed in the customer reviews section.		
Alternative Flows:	**N/A**		
Exceptions:	**12.0.E1 No completed orders.** Inform the customer that they must complete at least one order at the restaurant to leave a review. **12.0.E2 Missing star rating.** Inform the customer that a star rating is required.		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-35, BR-36, BR-37		
Other Information:	API Endpoint: /api/restaurant/{id}/rating		
Assumptions:	Reviews are visible to all users on the restaurant detail page.		

2.3.19 Follow restaurant

UC ID and Name:	**UC-13.1 Follow restaurant**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The customer clicks the “Theo dõi” button on a restaurant detail page.		
Description:	Allows customers to follow a restaurant so they can receive updates and easily access it from their followed list.		
Preconditions:	The customer must be logged in and must not already be following the restaurant.		
Postconditions:	The restaurant is added to the customer’s followed list.		
Normal flow:	**13.1 Follow restaurant.** 1. Customer visits the restaurant detail page. 2. Customer clicks the “Theo dõi” button. 3. The system adds the restaurant to the customer’s followed list. 4. The system updates the button to show “Đã theo dõi”.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Low		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	API Endpoint: /api/follow/add?restaurantId={id}		
Assumptions:	The system updates the follow status in real-time so that the customer sees the change immediately.		

2.3.20 Unfollow restaurant

UC ID and Name:	**UC-13.2 Unfollow restaurant**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The customer clicks the "Đã theo dõi" button on a restaurant they are currently following.		
Description:	Allows customers to unfollow a restaurant they have previously followed, removing it from their followed list.		
Preconditions:	The customer must be logged in and must currently be following the restaurant.		
Postconditions:	The restaurant is removed from the customer's followed list.		
Normal flow:	**13.2 Unfollow restaurant.** 1. Customer accesses the restaurant detail page or their list of followed restaurants. 2. Customer clicks the "Đã theo dõi" button. 3. The system removes the restaurant from the customer's followed list. 4. The system updates the interface to reflect the change.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Low		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	API Endpoint: /api/follow/unfollow?restaurantId={id}		
Assumptions:	The system updates the follow status in real-time so that the customer sees the change immediately.		

2.3.21 View profile customer

UC ID and Name:	**UC-14.1 View profile**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The Customer clicks the “Thông tin tài khoản” menu item in the profile dropdown.		
Description:	This use case allows users to view their personal information stored in the system.		
Preconditions:	The user is logged in.		
Postconditions:	Profile information is retrieved and displayed.		
Normal flow:	**14.1 View profile** 1. Customer accesses the profile section. 2. System retrieves user data. 3. System displays name, email, phone, avatar, birthday, etc…		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Regularly		
Business Rules:	N/A		
Other Information:	Endpoint: /api/customer/profile		
Assumptions:	Guest session is valid and the backend is available.		

2.3.22 Modify profile customer

UC ID and Name:	**UC-14.2 Modify profile**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The customer changes information on the Profile form and clicks “Cập nhật” button.		
Description:	This use case allows users to update their profile information (name, phone, avatar…).		
Preconditions:	Guest is logged in and viewing their own profile.		
Postconditions:	Profile information is updated in the system.		
Normal flow:	**14.2 Modify profile.** 1. Customer opens the Profile page. 2. Customer edits one or more fields (name, gender, phone, address, birthday). 3. Customer clicks the “Cập nhật” button. 4. The system validates the input. 5. The system saves the changes. 6. The system shows a success message and displays updated information.		
Alternative Flows:	N/A		
Exceptions:	**14.2.E1 Invalid phone/email format** Inform: “Thông tin không hợp lệ.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-59		
Other Information:	Endpoint: /api/customer/profile/update		
Assumptions:	Guest provides valid data and has permission to update their profile.		

2.3.23 Change password

UC ID and Name:	**UC-14.3 Change password**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The Customer clicks the “Đổi mặt khẩu” menu item in the profile dropdown.		
Description:	This use case allows users to change their current password.		
Preconditions:	Customer is logged in and authenticated.		
Postconditions:	Password is updated if validation is successful.		
Normal flow:	**14.3 Change password** 1. Customer enters an old password and a new password. 2. System verifies and updates.		
Alternative Flows:	N/A		
Exceptions:	**14.3.E1 Wrong old password** Inform: “Mật khẩu cũ không đúng.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-52		
Other Information:	Endpoint: /api/user/change-password		
Assumptions:	Customer knows the current password.		

2.3.24 Payment by banking

UC ID and Name:	**UC-15 Payment by banking**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	Casso, VietQR
Trigger:	The customer clicks “Thanh toán bằng QR” button on the reservation page.		
Description:	Allows the customer to complete payment for a reservation (deposit/hold fee) via QR bank transfer or other supported methods.		
Preconditions:	A reservation has been created and is in status Pending Payment; the payment time window is still valid.		
Postconditions:	If payment is successful, the reservation status becomes Paid/Confirmed; otherwise it remains Pending Payment or becomes Expired after timeout.		
Normal flow:	**15.0.A1 Pay from Reservation History (within 10 minutes)** 1. Customer opens "Reservation History". 2. The system displays a list of reservations that are in Pending Payment status and within 10 minutes of booking. 3. The customer click the “Thanh toán” button on the reservation to be paid. 4. Continue following Normal Flow steps 3–7 (create session, scan QR, confirm transaction, update Paid/Confirmed)..		
Alternative Flows:	N/A		
Exceptions:	**15.0.E1 Timeout.** No payment received within the window → show expired, keep reservation pending or release table. **15.0.E2 Underpaid/overpaid.** Inform customers and provide retry/refund guidance. **15.0.E3 Wrong content/reference.** Cannot map to reservation → inform customer, keep pending. **15.0.E4 Gateway/Casso error** Show error, allow retry; log detail.		
Priority:	High		
Frequency of Use:	Regularly		
Business Rules:	BR-08, BR-10, BR-15, BR-17, BR-28		
Other	Casso Polling: After a payment session is created, the backend runs. a Casso		

UC ID and Name:	**UC-15 Payment by banking**
Information: Other Information:	polling process continuously 30 seconds once time. Endpoint: /api/payments/status?reservationId={id} /api/reservations/{id}/mark-paid
Assumptions:	Webhook is preferred for near real-time confirmation; polling acts as a fallback. Late transactions are reconciled by periodic sync jobs.

2.3.25 View list of orders

UC ID and Name:	**UC-16.1 View list of orders**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The Customer clicks the “Theo dõi giao hàng” menu item in the profile dropdown.		
Description:	This use case allows users to view the list of food they have ordered for an active reservation.		
Preconditions:	The user has an active reservation or food order in the system.		
Postconditions:	The list of current ordered food items is displayed.		
Normal flow:	**16.1 View list of orders** 1. Guest accesses order section. 2. System fetches ordered food items. 3. System displays them.		
Alternative Flows:	N/A		
Exceptions:	**16.1.E1 No food orders found** Inform: “Bạn chưa đặt món nào.”		
Priority:	Medium		
Frequency of Use:	Regularly		
Business Rules:	N/A		
Other Information:	Endpoint: /api/order/index?customerId=...		
Assumptions:	Guest has placed at least one food order.		

2.3.26 Create order

UC ID and Name:	**UC-16.2 Create order**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer adds foods to cart and clicks “Mua Hàng” / “Xác nhận đơn hàng” button.		
Description:	Allows customers to create a food order from selected items in the cart, optionally applying a voucher, and submit the order to the restaurant.		
Preconditions:	Customer is logged in; cart has at least one food item from a restaurant; restaurant can accept orders.		
Postconditions:	A new order is created with status Placed/Pending; cart items used for the order are cleared (or marked as checked out).		
Normal flow:	**16.2 Create order.** 1. Customer clicks “Thêm vào giỏ” on a food item. 2. Customer opens “Giỏ hàng” and adjusts quantities if needed. 3. Customer clicks ”Mua Hàng” → system shows Checkout page with order summary. 4. Customer optionally selects or enters a voucher code; system recalculates totals. 5. Customer clicks “Xác nhận” đơn hàng. 6. The system validates stock/availability and prices, calculates total, shipping (if any), and voucher discount. 7. The system creates the order with status Placed/Pending and returns order detail/number; customer sees confirmation.		
Alternative Flows:	N/A		
Exceptions:	**16.2.E1 Out of stock** for one or more items → inform customer and block submission for those items. **16.2.E2 Restaurant not accepting orders** (closed/paused) → show message and prevent creation.		
Priority:	High		
Frequency of Use:	Frequently		
Business Rules:	BR-20, BR-21, BR-22, BR-23, BR-24, BR-25		

UC ID and Name:	**UC-16.2 Create order**
Other Information:	Endpoint: - Add/modify cart: /api/cart/items/{itemId} - Voucher check (optional): /api/voucher/{code} - Create order: /api/order - After success: clear checked-out items from cart.
Assumptions:	Orders are per-restaurant.

2.3.27 View history orders

UC ID and Name:	**UC-16.3 View history orders**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The Customer clicks the "Theo dõi giao hàng" menu item in the profile dropdown.		
Description:	This use case allows users to see past food orders for completed reservations		
Preconditions:	Customer is logged in.		
Postconditions:	Past food orders are listed by reservation or date.		
Normal flow:	**16.3 View history orders** 1. Customer clicks the history tab. 2. System loads completed orders. 3. System shows date, name, amount.		
Alternative Flows:	N/A		
Exceptions:	**16.3.E1 No order history found** Inform: "Bạn chưa có đặt món nào."		
Priority:	Low		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: /api/order/history		

UC ID and Name:	**UC-16.3 View history orders**
Assumptions:	Guests have made at least one food order in the past.

2.3.28 View list of posts

UC ID and Name:	**UC-17.1 View list of posts**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer selects “Mạng xã hội” link in the navigation bar.		
Description:	This use case allows users to view a list of their created posts.		
Preconditions:	Customer is logged in.		
Postconditions:	The list of posts created by the user is displayed.		
Normal flow:	**17.1 View posts** 1. Guest accesses “Mạng xã hội”. 2. System retrieves all posts created. 3. Posts are displayed.		
Alternative Flows:	N/A		
Exceptions:	**17.1.E1 No posts available** Inform: “HIện tại chưa có bài viết nào.”		
Priority:	Medium		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: /api/posts		
Assumptions:	Guest has at least one post.		

2.3.29 Create post

UC ID and Name:	**UC-17.2 Create post**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer clicks create post button (opens the create-post modal).		
Description:	Allows the customer to create a new post with text content and optionally attach images. The post is then published to their personal timeline.		
Preconditions:	Customer is logged in.		
Postconditions:	The post is saved in the system and displayed on the customer's timeline.		
Normal flow:	**17.2 Create Post** 1. Customer opens the create post modal. 2. Customer enters text content. 3. Customer clicks Add Image and selects one or more images. 4. Customer clicks Publish. 5. System validates the input (text/images). 6. System saves the post and any attached images. 7. System displays the new post on the timeline.		
Alternative Flows:	N/A		
Exceptions:	**17.2.E1 Empty content:** No text and no images → show error requiring text or image. **17.2.E2 Image upload failure / invalid format / exceeds size limit:** Show error and allow re-selection. **17.2.E3 Network/Server error:** Show error, keep entered content so the user can retry.		
Priority:	Medium		
Frequency of Use:	Frequently		
Business Rules:	BR-38, BR-39, BR-40, BR-41, BR-42, BR-59		

UC ID and Name:	**UC-17.2 Create post**
Other Information:	Endpoint: /api/posts/create. Upload image to S3 bucket in Amazon Web Service.
Assumptions:	Customer provides valid input. Images are stored in a S3 storage.

2.3.30 Modify post

UC ID and Name:	**UC-17.3 Modify post**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer clicks “Chỉnh sửa” button in specified post.		
Description:	This use case allows users to edit previously created posts.		
Preconditions:	Customer is logged in and owns the post.		
Postconditions:	The post is updated with new content.		
Normal flow:	**17.3 Modify post** 1. Guest clicks edit. 2. Customer loads post data. 3. Guests edit and save. 4. System updates post.		
Alternative Flows:	N/A		
Exceptions:	**17.3.E1 No post** System displays “No posts available.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-38, BR-39, BR-40, BR-41, BR-42, BR-59		
Other Information:	Endpoint: /api/posts/edit Old images need to be deleted and new images added to S3 bucket.		

UC ID and Name:	**UC-17.3 Modify post**
Assumptions:	The user is the author of the post.

2.3.31 Delete post

UC ID and Name:	**UC-17.4 Delete post**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The customer clicks “Xóa” button in a post.		
Description:	This use case allows users to remove their own posts from the system.		
Preconditions:	The customer is logged in and owns the post.		
Postconditions:	The post is permanently removed or marked as deleted.		
Normal flow:	**17.4 Delete post** 1. Customer clicks delete. 2. The system asks for confirmation. 3. Customer confirms. 4. The system deletes posts.		
Alternative Flows:	N/A		
Exceptions:	**17.4.E1 No post** System displays “No posts available.”		
Priority:	Medium		
Frequency of Use:	Sometimes		

UC ID and Name:	**UC-17.4 Delete post**
Business Rules:	BR-41, BR-42, BR-59
Other Information:	Endpoint: /api/posts/delete The image needs to be deleted in the S3 bucket.
Assumptions:	Customer is the author of the post.

2.3.32 View list of vouchers

UC ID and Name:	**UC-18.1 View list of vouchers**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer clicks on “Điểm thưởng” menu item from the account menu.		
Description:	This use case allows a customer to see all vouchers assigned to them, including status, expiry date, and discount detail.		
Preconditions:	Customer is logged in. Customer has at least one voucher assigned (optional).		
Postconditions:	Customer sees a list of their vouchers with relevant detail.		
Normal flow:	**18.1 View list of vouchers** 1. Actor navigates to the "My Vouchers" page. 2. System retrieves all vouchers assigned to the customer from the database. 3. System displays each voucher with detail.		
Alternative Flows:	N/A		
Exceptions:	**18.1.E1 No voucher** System displays “No vouchers available.”		

UC ID and Name:	**UC-18.1 View list of vouchers**
Priority:	Medium
Frequency of Use:	Frequent
Business Rules:	N/A
Other Information:	API endpoint (example): GET /api/vouchers/{cus_id}
Assumptions:	Customer has permission to view only their own vouchers.

2.3.33 View redeemable vouchers

UC ID and Name:	**UC-18.2 View redeemable voucher**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer clicks the “Đổi điểm” button in the My Voucher page.		
Description:	This use case allows a customer to view a list of vouchers that can be redeemed using their accumulated points.		
Preconditions:	Customer is logged in. Customer has a points balance (≥ 0).		
Postconditions:	Customer sees a list of redeemable vouchers.		
Normal flow:	**18.2 View redeemable vouchers** 1. Actor opens “My Voucher” page. 2. Actor clicks the “Đổi điểm” button. 3. System retrieves a list of vouchers available for redemption from the database. 4. System displays vouchers with the following detail. 5. Actor reviews vouchers and can choose one to redeem.		

UC ID and Name:	**UC-18.2 View redeemable voucher**
Alternative Flows:	N/A
Exceptions:	**18.2.E1 No redeemable vouchers available** System shows message “Hiện chưa có voucher nào để đổi.”
Priority:	Medium
Frequency of Use:	Sometimes
Business Rules:	N/A
Other Information:	API endpoint (example): GET /api/vouchers/redeemable
Assumptions:	Customer has points in the loyalty/reward system.

2.3.34 Redeem point

UC ID and Name:	**UC-18.3 Redeem point**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer selects a redeemable voucher and clicks “Đổi điểm” button.		
Description:	This use case allows a customer to exchange their reward points for a voucher.		
Preconditions:	Customer is logged in. Customer has enough points to redeem the selected voucher. Voucher is still available (not expired, not out of stock).		
Postconditions:	Customer points are deducted. Voucher is assigned to the customer’s account and appears in “My Voucher”.		
Normal flow:	**18.3 Redeem point** 1. Actor navigates to the “Redeemable Voucher” list. 2. Actor clicks “Đổi điểm” on a specific voucher.		

UC ID and Name:	**UC-18.3 Redeem point**
	3. System checks the following: Actor's point balance ≥ required points. Voucher is available and valid. 4. System deducts the required points from the customer's balance. 5. System assigns the voucher to the customer's account. 6. System shows a success message. 7. Actor can go to "My Voucher" to view the newly redeemed voucher.
Alternative Flows:	N/A
Exceptions:	**18.3.E1 Insufficient points** System shows message "Bạn không đủ điểm để đổi voucher này." **18.3.E2 Voucher out of stock** System shows message "Voucher này đã hết lượt đổi."
Priority:	High
Frequency of Use:	Sometimes
Business Rules:	BR-30, BR-31
Other Information:	API endpoint (example): POST /api/vouchers/redeem
Assumptions:	Customer has points in the loyalty/reward system.

2.3.35 Give voucher

UC ID and Name:	**UC-18.4 Give voucher**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer clicks “Tặng voucher” button on a voucher in My Voucher page.		
Description:	This use case allows a customer to gift an unused voucher to a friend who is already in their friend list.		
Preconditions:	Customer is logged in. Voucher is unused and valid (not expired). Recipient is in the customer’s friend list.		
Postconditions:	Voucher ownership is transferred from the giver to the recipient. The recipient sees the voucher in their “My Voucher” list. Giver no longer has that voucher.		
Normal flow:	**18.4 Give voucher** 1. Actor goes to the “My Voucher” page. 2. Actor clicks “Tặng voucher” button on a specific voucher.		

UC ID and Name:	**UC-18.4 Give voucher**
	3. System shows a modal or popup with the friend list. 4. Actor selects a friend and confirms the action. 5. System checks voucher validity (unused, not expired). 6. System transfers the voucher to the selected friend. 7. Recipient sees the voucher in their “My Voucher” page.
Alternative Flows:	N/A
Exceptions:	**18.4.E1 No friends in the list** Show message “Bạn chưa có bạn bè để tặng voucher.” **18.4.E2 Voucher already used or expired** Show error message.
Priority:	Medium
Frequency of Use:	Sometimes
Business Rules:	BR-34
Other Information:	API endpoint (example): POST /api/vouchers/give
Assumptions:	Both giver and receiver are active customers. Friend relationships exist in the system before gifting.

2.3.36 View redemption history

UC ID and Name:	**UC-18.5 View redemption history**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer clicks “Lịch sử” button on the My Voucher page.		
Description:	This use case allows a customer to view their history of redeeming points for vouchers.		
Preconditions:	Customer is logged in. Customer has at least one redeem history record (optional).		
Postconditions:	Customer sees a list of past voucher redemption transactions.		

UC ID and Name:	**UC-18.5 View redemption history**
Normal flow:	**18.5 View redemption history** 1. Actor opens “My Voucher” page. 2. Actor clicks “Lịch sử” button. 3. System retrieves customer’s redemption history from the database. 4. System displays history records with detail
Alternative Flows:	N/A
Exceptions:	**18.5.E1 No redemption history** System shows message “Bạn chưa có lịch sử đổi điểm nào.”
Priority:	Medium
Frequency of Use:	Sometimes
Business Rules:	N/A
Other Information:	API endpoint (example): GET /api/vouchers/redeem/history/{customer_id}
Assumptions:	Customer has a points system account.

2.3.37 Add to cart

UC ID and Name:	**UC-19 Add to cart**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Guest clicks “Thêm vào giỏ” button in food page.		
Description:	This use case allows users to add food items to their shopping cart.		
Preconditions:	Guest is logged in and on the food menu page.		

UC ID and Name:	**UC-19 Add to cart**
Postconditions:	Selected item is added to cart.
Normal flow:	**19.0 Add to cart** 1. Guest clicks add. 2. System updates cart and total.
Alternative Flows:	N/A
Exceptions:	**19.0.E1 Item unavailable** Inform: “Món ăn này hiện không có.”
Priority:	High
Frequency of Use:	Frequently
Business Rules:	BR-18
Other Information:	Endpoint: /api/cart/addToCart
Assumptions:	Food items are valid and in stock.

2.3.38 Register to be restaurant owner

UC ID and Name:	**UC-20 Register to be restaurant owner**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer selects “Tạo chủ nhà hàng” button in profile menu.		

UC ID and Name:	**UC-20 Register to be restaurant owner**
Description:	This use case allows users to create restaurant owners to manage restaurants.
Preconditions:	The customer is logged in and authenticated.
Postconditions:	Registration request is submitted.
Normal flow:	**20.0 Register restaurant owner** 1. Customer fills name of restaurant owner. 2. System save.
Alternative Flows:	N/A
Exceptions:	**20.0.E1 Already registered** Inform: “Bạn đã đăng ký trước đó.”
Priority:	Medium
Frequency of Use:	Sometimes
Business Rules:	BR-53
Other Information:	Endpoint: /api/restaurant/register-owner
Assumptions:	Customer provides valid business and contact information.

2.3.39 Chat with restaurant owner

UC ID and Name:	**UC-21 Chat with restaurant owner**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025

UC ID and Name:	**UC-21 Chat with restaurant owner**		
Primary actor:	Customer	Secondary Actors:	Restaurant Owner
Trigger:	Customer opens the Chat panel and selects a restaurant to start/continue a conversation.		
Description:	Allows a customer to exchange messages with a restaurant through a real-time chat window. Chat is permitted only if the customer is following that restaurant.		
Preconditions:	Customer is logged in. Customer follows the selected restaurant.		
Postconditions:	A chat thread exists between the customer and the restaurant; messages are stored and visible in both parties' inboxes.		
Normal flow:	**21.0 Start and send messages.** 1. Customer opens the Chat panel → "Restaurants" tab. 2. System lists restaurants the customer follows. 3. Customer selects a restaurant; if no thread exists, the system creates one. 4. System opens the chat window and loads recent messages. 5. Customer types a message and clicks Send. 6. System delivers the message (real-time) and persists. 7. Restaurant can reply; conversation updates live.		
Alternative Flows:	N/A		
Exceptions:	**21.E1 Restaurant unavailable/offline.** The system queues the message and marks as not response yet.		
Priority:	Medium		
Frequency of Use:	Regularly		
Business Rules:	BR-55		
Other Information:	Endpoint: /api/chat/restaurant		
Assumptions:	Push notifications are sent to the restaurant's dashboard/app; typing indicators and read receipts are optional enhancements.		

2.3.40 Chat with AI assistant

UC ID and Name:	**UC-22 Chat with AI assistant**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Guest	Secondary Actors:	AI Assistant API
Trigger:	Guests click on the Virtual Assistant icon in the bottom right corner of the screen.		
Description:	This use case allows users to interact with an AI assistant to get help or suggestions.		
Preconditions:	The system is online and accessible.		
Postconditions:	The AI assistant replies with an appropriate response.		
Normal flow:	**22.0 Chat with AI assistant** 1. Guest asks a question. 2. AI receives and responds.		
Alternative Flows:	**22.1 AI cannot answer** Inform: “Tôi chưa có thông tin về câu hỏi này.”		
Exceptions:	**22.0.E1 API timeout** Inform: “Hệ thống bận, vui lòng thử lại.”		
Priority:	High		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: /api/chat/assistant		
Assumptions:	AI systems are trained and online.		

2.3.41 Logout in customer page

UC ID and Name:	**UC-23 Logout**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer clicks “Đăng xuất” link in profile menu.		
Description:	This use case allows users to securely log out from the application.		
Preconditions:	Customer is logged in and authenticated.		
Postconditions:	Customer session is destroyed, and the system returns to the home or login screen.		
Normal flow:	**23.0 Logout** 1. Customer clicks logout. 2. System clears session and redirects.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: /api/auth/logout		
Assumptions:	Customer intends to leave the system securely.		

2.3.42 View list of RO’s restaurants

UC ID and Name:	**UC-24.1 View list of RO's restaurants**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks "Danh sách nhà hàng" menu item in manage menu.		
Description:	This use case allows restaurant owners to view a list of their registered restaurants.		
Preconditions:	Owner is logged in and has at least one registered restaurant.		
Postconditions:	A list of all restaurants created by the owner is displayed.		
Normal flow:	**24.1 View list of RO's restaurants** 1. Owner accesses the restaurant owner management page. 2. System loads restaurants. 3. Owner views them.		
Alternative Flows:	N/A		
Exceptions:	**24.1.E1 No restaurants found** Inform: "Bạn chưa đăng ký nhà hàng nào."		
Priority:	Medium		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: /api/restaurant/index		
Assumptions:	Owner has valid access rights.		

2.3.43 Create RO's restaurant

UC ID and Name:	**UC-24.2 Create RO's restaurant**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks "Thêm nhà hàng" button in manage restaurant page.		
Description:	This use case allows owners to register a new restaurant with system detail.		
Preconditions:	Owner is logged in.		
Postconditions:	A new restaurant is created and added to the system.		
Normal flow:	**24.2 Create RO's restaurant** 1. Owner fills out the form. 2. Upload image, information of restaurant. 3. Submits. 4. System stores data.		
Alternative Flows:	N/A		
Exceptions:	**24.2.E1 Missing required fields** Inform: "Vui lòng nhập đầy đủ thông tin."		
Priority:	High		
Frequency of Use:	Sometimes		
Business Rules:	BR-01, BR-59		
Other Information:	Endpoint: /api/restaurant/create		
Assumptions:	Owner provides all required data.		

2.3.44 Modify RO's restaurant

UC ID and Name:	**UC-24.3 Modify RO's restaurant**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks "Cập nhật nhà hàng" button in restaurant.		
Description:	This use case allows owners to update their restaurant information.		
Preconditions:	Owner is logged in and owns the restaurant.		
Postconditions:	The restaurant detail are updated in the system.		
Normal flow:	**24.3 Modify RO's restaurant** 1. Owner selects a restaurant. 2. Updates info. 3. Saves changes. 4. System updates record.		
Alternative Flows:	N/A		
Exceptions:	**24.3.E1 Invalid data** Inform: "Dữ liệu nhập không hợp lệ."		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-01, BR-59		
Other Information:	Endpoint: /api/restaurant/edit		
Assumptions:	Owner has editing rights.		

2.3.45 Close RO's restaurant

UC ID and Name:	**UC-24.4 Close RO's restaurant**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks "Kích hoạt nhà hàng" button in edit restaurant page.		
Description:	This use case allows owners to close (deactivate) their restaurant, making it unavailable to customers.		
Preconditions:	Owner is logged in and owns the restaurant.		
Postconditions:	The restaurant status is updated to "Inactive" in the system and is hidden from customers.		
Normal flow:	**24.4 Close RO's restaurant** 1. Owner selects a restaurant. 2. Opens update info page. 3. Disables the active status. 4. Saves changes. 5. System updates record to inactive.		
Alternative Flows:	N/A		
Exceptions:	**24.4.E1 Cannot inactive restaurant with reservations** Inform: "Không thể đóng nhà hàng đang được sử dụng." **24.4.E2 System error** Inform: "Không thể đóng nhà hàng. Vui lòng thử lại."		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-60		
Other Information:	Endpoint: /api/restaurant/edit		
Assumptions:	Owner has editing rights for the restaurant.		

2.3.46 Submit business license

UC ID and Name:	**UC-24.5 Submit business license**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks "+ Đăng ký giấy phép mới" button in the business license list page.		
Description:	This use case allows the owner to submit a business license for their restaurant.		
Preconditions:	The owner is logged in and owns at least one restaurant.		
Postconditions:	The submitted business license is saved in the system and marked as pending verification.		
Normal flow:	**24.5 Submit business license** 1. Owner opens the business license list page. 2. Owner clicks "+ Đăng ký giấy phép mới". 3. Owner selects a restaurant. 4. Owner enters tax code. 5. Owner uploads related documents. 6. Owner uploads license file (PDF/JPG/PNG). 7. Owner clicks "Đăng ký". 8. System saves the license and marks it as pending verification.		
Alternative Flows:	N/A		
Exceptions:	**24.5.E1 Missing or invalid data** Inform: "Dữ liệu nhập không hợp lệ hoặc thiếu tệp giấy phép."		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-57		
Other Information:	Endpoint: /api/restaurant/license/submit		
Assumptions:	Owner has permission to submit a license for the restaurant.		

2.3.47 View list of categories

UC ID and Name:	**UC-25 View list of categories**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner go to the create or edit food page.		
Description:	This use case allows owner to view all available categories.		
Preconditions:	Owner is logged in.		
Postconditions:	List of categories is displayed.		
Normal flow:	**25.1 View categories** 1. Actor accesses the create or edit food page. 2. System loads categories. 3. Actor views them.		
Alternative Flows:	N/A		
Exceptions:	**25.1.E1 No categories found** Inform: “Hiện chưa có danh mục nào.”		
Priority:	Medium		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/category		
Assumptions:	Actor has full permission to view all categories.		

2.3.48 View list of areas

UC ID and Name:	**UC-26.1 View list of areas**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Selects “Danh sách khu vực” menu item from the manage menu.		
Description:	This use case allows the actor to view all areas belonging to their restaurant.		
Preconditions:	Owner is logged in; the restaurant context is selected.		
Postconditions:	List of areas is displayed.		
Normal flow:	**26.1 View areas** 1. Owner accesses the area list. 2. The system loads areas for the restaurant. 3. Owner reviews them.		
Alternative Flows:	N/A		
Exceptions:	**26.1.E1 No areas found** → Inform: “Hiện chưa có khu vực nào.”		
Priority:	Medium		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/areas?restaurantId={id}		
Assumptions:	Actor has permission to view all areas of their restaurant.		

2.3.49 Create areas

UC ID and Name:	**UC-26.2 Create area**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Clicks create an area button on the manage area page.		
Description:	This use case allows the actor to add a new area to the restaurant.		
Preconditions:	Actor is logged in; Areas page is open.		
Postconditions:	A new area is saved and appears in the list.		
Normal flow:	**26.2 Create an area** 1. Actor opens the create form. 2. Actor enters Area Name (required) and optional Description/Note. 3. System validates inputs. 4. System saves the new area. 5. System shows success and updates the list.		
Alternative Flows:	N/A		
Exceptions:	**26.2.E1 Validation failed** Show specific field errors. **26.2.E2 Duplicate name** Inform: “Tên khu vực đã tồn tại trong nhà hàng.”		
Priority:	High		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: POST /api/areas		
Assumptions:	Actor has permission to manage areas for their restaurant.		

2.3.50 Modify area

UC ID and Name:	**UC-26.3 Modify area**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Clicks edit button on a specific area.		
Description:	This use case allows the actor to update an existing area's information.		
Preconditions:	Actor is logged in; target area exists and belongs to the restaurant.		
Postconditions:	Area detail are updated and displayed.		
Normal flow:	**26.3 Modify area** 1. Actor clicks edit on a specific area. 2. System displays current area detail. 3. Actor modifies information. 4. System checks if any tables in the area are in use. 5. If no tables are in use, System validates inputs. 6. System updates the area in the database. 7. System displays success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**26.3.E1 Area not found** Inform and return to list. **26.3.E2 Duplicate name** Inform: "Tên khu vực đã tồn tại." **26.3.E3 Concurrency conflict** Inform: "Khu vực đã được cập nhật bởi người khác, vui lòng tải lại." **26.3.E4 Area in use** Inform: "Không thể cập nhật khu vực khi đang có bàn được sử dụng."		
Priority:	High		
Frequency of Use:	Sometimes		
Business Rules:	N/A		

UC ID and Name:	**UC-26.3 Modify area**
Other Information:	Endpoint: PUT /api/areas/{areaId}
Assumptions:	Actor is authorized to edit areas in their restaurant.

2.3.51 Delete area

UC ID and Name:	**UC-26.4 Delete area**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Clicks delete button on a specific area and confirms.		
Description:	This use case allows the actor to remove an existing area from the restaurant.		
Preconditions:	Actor is logged in; area exists and belongs to the restaurant.		
Postconditions:	Area is deleted or kept (if deletion is blocked).		
Normal flow:	**26.4 Delete area** 1. Actor clicks delete on a specific area. 2. System checks if any tables in the area are in use. 3. If no tables are in use, System asks for confirmation. 4. Actor confirms deletion. 5. System removes the area. 6. System displays success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**26.4.E1 Area in use** Inform: “Không thể xoá khu vực khi đang có bàn được sử dụng.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: DELETE /api/areas/{areaId}		

UC ID and Name:	**UC-26.4 Delete area**
Assumptions:	Actor has delete permission; referential integrity is enforced.

2.3.52 View list of tables

UC ID and Name:	**UC-27.1 View list of tables**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks “Danh sách bàn” menu item in the management restaurant page.		
Description:	This use case allows owners to see all tables for their restaurant.		
Preconditions:	Owner is logged in and has at least one restaurant.		
Postconditions:	System shows a table list.		
Normal flow:	**27.1 View tables** 1. Owner accesses the table list. 2. System retrieves and displays data.		
Alternative Flows:	N/A		
Exceptions:	**27.1.E1 No tables found** Inform: “Bạn chưa thêm bàn nào.”		
Priority:	Medium		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: /api/restaurant/table		

UC ID and Name:	**UC-27.1 View list of tables**
Assumptions:	Tables exist in the restaurant system.

2.3.53 Create table

UC ID and Name:	**UC-27.2 Create table**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	The owner clicks the add table button from the list table page.		
Description:	This use case allows owners to add a new table to their restaurant.		
Preconditions:	Owner is logged in and has a selected restaurant.		
Postconditions:	New table is saved and available for reservations.		
Normal flow:	**27.2 Create table** 1. Owner fills info (name, seat, image). 2. Submits. 3. System adds to list.		
Alternative Flows:	N/A		
Exceptions:	**27.2.E1 Missing fields** Inform: “Vui lòng điền đầy đủ thông tin.” **26.2.E2 Duplicate name** Inform: “Tên bàn đã tồn tại.”		
Priority:	High		
Frequency of Use:	Sometimes		

UC ID and Name:	**UC-27.2 Create table**
Business Rules:	BR-59
Other Information:	Endpoint: /api/restaurant/table/create
Assumptions:	Owner has access to the restaurant.

2.3.54 Modify table

UC ID and Name:	**UC-27.3 Modify table**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks on a table of the list to edit.		
Description:	This use case allows owners to modify table information.		
Preconditions:	Table exists and belongs to the logged-in owner.		
Postconditions:	Table data is updated.		
Normal flow:	**27.3 Modify table** 1. Actor clicks [Edit] on a specific table. 2. System displays current table detail. 3. Actor modifies information (e.g., Name, Seats, Area). 4. System validates inputs. 5. System updates the table in the database. 6. System displays success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**27.3.E1 Table not found** Inform and return to list. **27.3.E2 Duplicate table name in the same area** Inform: "Tên bàn đã tồn tại trong khu vực này."		

UC ID and Name:	**UC-27.3 Modify table**
	27.3.E3 Concurrency conflict Inform: "Bàn đã được cập nhật bởi người khác, vui lòng tải lại."
Priority:	High
Frequency of Use:	Sometimes
Business Rules:	BR-59
Other Information:	Endpoint: PUT /api/tables/{tableId}
Assumptions:	Actor is authorized to edit tables in their restaurant.

2.3.55 Delete tables

UC ID and Name:	**UC-27.4 Delete table**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks “Xóa” button at the table of the restaurant.		
Description:	This use case allows restaurant owners to delete tables from their list.		
Preconditions:	Table exists and is not linked to a future reservation.		
Postconditions:	The table is removed from the system.		
Normal flow:	**27.4 Delete table** 1. Actor clicks [Delete] on a specific table. 2. System asks for confirmation. 3. Actor confirms deletion. 4. System removes the table from the database. 5. System displays success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**27.4.E1 Table not found** Inform and refresh list.		

UC ID and Name:	**UC-27.4 Delete table**
	27.4.E2 Table in use (reserved or occupied) Inform: "Không thể xoá bàn khi đang được đặt hoặc sử dụng." **27.4.E3 Server error** Show generic error.
Priority:	Medium
Frequency of Use:	Sometimes
Business Rules:	N/A
Other Information:	Endpoint: DELETE /api/tables/{tableId}
Assumptions:	Actor has delete permission; referential integrity is enforced.

2.3.56 View list of customer reservations

UC ID and Name:	**UC-28.1 View list of customer reservations**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks “Danh sách đặt bàn” menu item in manage page.		
Description:	This use case allows restaurant owners to see all incoming reservation requests.		
Preconditions:	Owner is logged in and owns a restaurant.		
Postconditions:	A list of reservations is displayed.		
Normal flow:	**28.1 View list of customer reservations** 1. Actor accesses the reservation list page. 2. System loads all reservations based on selected filters (area, status). 3. Actor views the reservations.		
Alternative Flows:	N/A		

UC ID and Name:	**UC-28.1 View list of customer reservations**
Exceptions:	**28.1.E1 No reservations found** Inform: “HIện chưa có bàn nào.”
Priority:	Medium
Frequency of Use:	Frequently
Business Rules:	BR-09
Other Information:	Endpoint: /api/reservation
Assumptions:	Actor has permission to view all reservations of their restaurant.

2.3.57 Create reservation

UC ID and Name:	**UC-28.2 Create reservation**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner chooses available time on a specific table and click “Đặt bàn” button.		
Description:	This use case allows restaurant owners to create a reservation for a customer.		
Preconditions:	Actor is logged in; table exists and is available.		
Postconditions:	A new reservation is created and linked to the table.		
Normal flow:	**28.2 Create reservation** 1. Actor clicks "Đặt bàn" for a specific table. 2. System displays the reservation creation form. 3. Actor enters reservation detail. 4. System validates inputs.		

UC ID and Name:	**UC-28.2 Create reservation**
	5. System saves the reservation. 6. System displays success message and updates the table status.
Alternative Flows:	N/A
Exceptions:	**28.2.E1 Validation failed** Show error message. **28.2.E2 Table already reserved** Inform: "Bàn đã được đặt."
Priority:	High
Frequency of Use:	Sometimes
Business Rules:	BR-02, BR-03, BR-04, BR-05, BR-06, BR-07, BR-11, BR-16
Other Information:	Endpoint: POST /api/reservations
Assumptions:	Actor has permission to create reservations.

2.3.58 Modify reservation

UC ID and Name:	**UC-28.3 Modify reservation**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks the “Edit” icon on a specific reservation row in the Reservation Management dashboard.		
Description:	This use case allows the restaurant owner to modify detail of an existing reservation, such as reservation status or assigned table.		
Preconditions:	Actor is logged in; reservation exists.		
Postconditions:	If reservation exists and input is valid → reservation detail are updated in the database and reflected in the list. If reservation does not exist → system shows error message and returns to the list.		

UC ID and Name:	**UC-28.3 Modify reservation**
Normal flow:	**28.3 Modify reservations** 1. Actor clicks edit button on a reservation. 2. System displays current reservation detail. 3. Actor edits the status table. 4. System validates inputs. 5. System updates the reservation in the database. 6. System displays success message and refreshes the list.
Alternative Flows:	N/A
Exceptions:	**28.3.E1 Reservation not found** Inform and return to list. **28.3.E2 Validation failed** Show error message.
Priority:	High
Frequency of Use:	Sometimes
Business Rules:	BR-09, BR-12, BR-13, BR-14
Other Information:	Endpoint: PUT /api/reservations/{reservationId}
Assumptions:	Actor has permission to modify reservations.

2.3.59 View customer reservations history

UC ID and Name:	**UC-28.4 View customer reservations history**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks “Lịch sử đặt bàn” menu item in manage page.		
Description:	This use case allows owners to view all past reservations.		
Preconditions:	Owner is logged in.		
Postconditions:	Past reservations are displayed with status.		

UC ID and Name:	**UC-28.4 View customer reservations history**
Normal flow:	**28.4 View customer reservations history** 1. Owner selects history. 2. System fetches all completed/canceled reservations.
Alternative Flows:	N/A
Exceptions:	**28.4.E1 No past reservations** Inform: “Không có lịch sử đặt bàn.”
Priority:	Low
Frequency of Use:	Sometimes
Business Rules:	BR-09
Other Information:	Endpoint: GET /api/reservation/history
Assumptions:	Actor has permission to view reservation history.

2.3.60 View list of menus

UC ID and Name:	**UC-29.1 View list of menus**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner selects "Quản lí menu" menu item in the manage page.		
Description:	This use case allows restaurant owners to view all menus of their restaurant.		

UC ID and Name:	**UC-29.1 View list of menus**
Preconditions:	Actor is logged in; restaurant exists.
Postconditions:	The list of menus is displayed.
Normal flow:	**29.1 View list of menus** 1. Actor accesses the menu list page. 2. System loads all menus for the restaurant. 3. Actor views the menu list.
Alternative Flows:	N/A
Exceptions:	**29.1.E1 No menus found** Inform: "Chưa có menu nào."
Priority:	Medium
Frequency of Use:	Frequently
Business Rules:	N/A
Other Information:	Endpoint: GET /api/menus?restaurantId={id}
Assumptions:	Actor has permission to view all menus of their restaurant.

2.3.61 Create menu

UC ID and Name:	**UC-29.2 Create menu**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None

UC ID and Name:	**UC-29.2 Create menu**
Trigger:	Owner clicks "Thêm Menu" button on the menu list page.
Description:	This use case allows restaurant owners to create a new menu for their restaurant.
Preconditions:	Owner is logged in.
Postconditions:	A new menu is created and displayed in the menu list.
Normal flow:	**29.2 Create menu** 1. Actor clicks "Thêm Menu". 2. System displays the create menu form. 3. Actor enters menu detail (name, type, image). 4. System validates inputs. 5. System saves the new menu. 6. System displays success message and updates the list.
Alternative Flows:	N/A
Exceptions:	**29.2.E1 Validation failed** Show error message. **29.2.E2 Duplicate menu name** Inform: "Tên menu đã tồn tại."
Priority:	High
Frequency of Use:	Sometimes
Business Rules:	BR-59
Other Information:	Endpoint: POST /api/menus
Assumptions:	Actor has permission to create menus.

2.3.62 Modify menu

UC ID and Name:	**UC-29.3 Modify menu**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks "Sửa" button on a specific menu.		
Description:	This use case allows restaurant owners to modify an existing menu.		
Preconditions:	Actor is logged in; menu exists and belongs to the restaurant.		
Postconditions:	Menu detail are updated in the system.		
Normal flow:	**29.3 Modify menu** 1. Actor clicks "Sửa" button on a specific menu. 2. System displays current menu detail. 3. Actor modifies the menu information. 4. System validates inputs. 5. System updates the menu in the database. 6. System displays success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**29.3.E1 Menu not found** Inform and return to list. **29.3.E2 Validation failed** Show error message. **29.3.E3 Duplicate menu name** Inform: "Tên menu đã tồn tại."		
Priority:	High		
Frequency of Use:	Sometimes		
Business Rules:	BR-59		
Other Information:	Endpoint: PUT /api/menus/{menuId}		
Assumptions:	Actor has permission to modify menus.		

2.3.63 Delete menu

UC ID and Name:	**UC-29.4 Delete menu**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks "Xóa" button on a specific menu and confirms deletion.		
Description:	This use case allows restaurant owners to delete a menu.		
Preconditions:	Actor is logged in; menu exists and belongs to the restaurant.		
Postconditions:	Menu is removed from the system.		
Normal flow:	**29.4 Delete menu** 1. Actor clicks "Xóa" button on a specific menu. 2. System asks for confirmation. 3. Actor confirms deletion. 4. System deletes the menu from the database. 5. System displays success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**29.4.E1 Menu not found** Inform and refresh list. **29.4.E2 Menu in use (linked to existing foods)** Inform: "Không thể xóa menu khi còn món ăn thuộc menu."		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-61		
Other Information:	Endpoint: DELETE /api/menus/{menuId}		
Assumptions:	Actor has delete permission; referential integrity is enforced.		

2.3.64 View list of foods

UC ID and Name:	**UC-30.1 View list of foods**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner selects “Quản lý món” button on the specified menu.		
Description:	This use case allows restaurant owners to view all foods under a selected menu.		
Preconditions:	Actor is logged in; menu exists and belongs to the restaurant.		
Postconditions:	The list of foods is displayed.		
Normal flow:	**30.1 View list of foods** 1. Actor opens the food list of a selected menu. 2. System loads foods for that menu. 3. Actor views the list.		
Alternative Flows:	N/A		
Exceptions:	**30.1.E1 No foods available** Inform: “Bạn chưa có món ăn nào.”		
Priority:	Medium		
Frequency of Use:	Frequently		
Business Rules:	BR-45		
Other Information:	Endpoint: GET /api/foods?menuId={menuId}		
Assumptions:	Actor has permission to view foods of the selected menu.		

2.3.65 Create food

UC ID and Name:	**UC-30.2 Create food**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks “Thêm món ăn” button on the food page.		
Description:	This use case allows restaurant owners to create a new food item in a selected menu.		
Preconditions:	Actor is logged in; target menu exists and belongs to the restaurant.		
Postconditions:	A new food is created and appears in the list.		
Normal flow:	**30.2 Create food** 1. Actor clicks “Thêm món” button. 2. System displays the create food form. 3. Actor enters food detail (name, description, price, status, image, tags). 4. System validates inputs. 5. System saves the new food and links it to the menu. 6. System displays success message and updates the list.		
Alternative Flows:	N/A		
Exceptions:	**30.2.E1 Validation failed** Show field errors. **30.2.E2 Duplicate food name in the same menu** Inform: “Tên món đã tồn tại trong menu này.”		
Priority:	High		
Frequency of Use:	Frequently		
Business Rules:	BR-44, BR-59		
Other Information:	Endpoint: POST /api/foods		
Assumptions:	Actor has permission to create foods for the selected menu.		

2.3.66 Modify food

UC ID and Name:	**UC-30.3 Modify food**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks “Sửa” button on a specific food.		
Description:	This use case allows restaurant owners to edit information of an existing food item.		
Preconditions:	Actor is logged in; food exists and belongs to the selected menu/restaurant.		
Postconditions:	Food detail are updated and displayed.		
Normal flow:	**30.3 Modify food** 1. Actor clicks “Sửa” on a food item. 2. System displays current food detail. 3. Actor updates fields (name, description, price, status, image, tags). 4. System validates inputs. 5. System saves changes to the database. 6. System displays success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**30.3.E1 Food not found** Inform and return to list. **30.3.E2 Duplicate food name in the same menu** Inform: “Tên món đã tồn tại trong menu này.” **30.3.E3 Validation failed** Show field errors. **30.3.E4 Concurrency conflict** Inform: “Món ăn đã được cập nhật bởi người khác, vui lòng tải lại.”		
Priority:	High		
Frequency of Use:	Sometimes		
Business Rules:	BR-44, BR-45, BR-59		
Other Information:	Endpoint: PUT /api/foods/{foodId}		
Assumptions:	Actor has permission to modify foods.		

2.3.67 Delete food

UC ID and Name:	**UC-30.4 Delete food**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks “Xóa” button on a specific food and confirms.		
Description:	This use case allows restaurant owners to delete a food item from a menu.		
Preconditions:	Actor is logged in; food exists and belongs to the restaurant.		
Postconditions:	Food is removed from the system or kept if deletion is blocked..		
Normal flow:	**30.4 Delete food** 1. Actor clicks “Xóa” button on a food item. 2. System asks for confirmation. 3. Actor confirms deletion. 4. System deletes the food from the database. 5. System displays success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**30.4.E1 Food in active use** Inform: “Không thể xóa món ăn đang có đơn hàng.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-62		
Other Information:	Endpoint: DELETE /api/foods/{foodId}		
Assumptions:	Actor has delete permission for foods of the selected menu.		

2.3.68 View list of customer orders

UC ID and Name:	**UC-31.1 View list of customer orders**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks "Danh sách đơn món ăn" menu item in the restaurant management panel.		
Description:	This use case allows restaurant owners to view the list of current food orders.		
Preconditions:	Restaurant owner is logged in and has access to the restaurant's management panel.		
Postconditions:	The system displays the list of food orders with relevant detail.		
Normal flow:	**31.1 View customer orders list** 1. Actor clicks [Danh sách đơn món ăn]. 2. System retrieves the list of current orders from the database. 3. System displays the list with order detail (order ID, customer name, items, quantity, total price, status).		
Alternative Flows:	N/A		
Exceptions:	**31.1.E1 No orders found** Inform: “Không có đơn món ăn nào."		
Priority:	High		
Frequency of Use:	Regularly		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/orders/foods		
Assumptions:	Actor has view permission.		

2.3.69 Modify status customer order

UC ID and Name:	**UC-31.2 Modify status customer order**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner selects an order and clicks edit button.		
Description:	This use case allows restaurant owners to update the status of a food order.		
Preconditions:	Order exists and is linked to the owner's restaurant.		
Postconditions:	The order status is updated in the system.		
Normal flow:	**31.2 Modify status customer food** 1. Actor selects an order from the list. 2. Actor clicks edit button. 3. System asks for new status selection. 4. Actor selects a new status and confirms. 5. System updates the order status in the database. 6. System displays success message and refreshes the list..		
Alternative Flows:	N/A		
Exceptions:	**31.2.E1 Order not found** Inform and refresh list.		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-26		
Other Information:	Endpoint: PUT /api/orders/foods/{orderId}/status		
Assumptions:	Actor has edit permission; referential integrity is enforced.		

2.3.70 View history customer orders

UC ID and Name:	**UC-31.3 View history customer orders**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks "Lịch sử đơn món ăn" button in the restaurant management panel.		
Description:	This use case allows restaurant owners to view the history of completed or canceled food orders.		
Preconditions:	Restaurant owner is logged in and has access to the restaurant's management panel.		
Postconditions:	The system displays the list of historical food orders.		
Normal flow:	**31.3 View history orders** 1. Actor clicks "Lịch sử đơn món ăn" button. 2. System retrieves the list of completed or canceled orders from the database. 3. System displays the history list with relevant detail (order ID, customer name, items, quantity, total price, status, date).		
Alternative Flows:	N/A		
Exceptions:	**31.3.E1 No history orders found** Inform: "Không có lịch sử đơn món ăn."		
Priority:	Low		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/orders/foods/history		
Assumptions:	Actor has view permission.		

2.3.71 View revenue by reservations

UC ID and Name:	**UC-32 View revenue by reservations**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks “Doanh thu từ đặt bàn” menu item in manage page.		
Description:	This use case allows restaurant owners to view total revenue generated from reservations.		
Preconditions:	Owner is logged in.		
Postconditions:	A revenue chart or list by reservation is shown.		
Normal flow:	**32.0 View reservation revenue** 1. Owner opens the revenue tab. 2. System loads total reservation revenue. 3. Displays amount and summary.		
Alternative Flows:	N/A		
Exceptions:	**32.0.E1 No data** Inform: “Không có dữ liệu doanh thu từ đặt bàn.”		
Priority:	High		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: /api/revenue/reservation		
Assumptions:	Reservations have valid payment completed.		

2.3.72 Filter by monthly revenue of reservations

UC ID and Name:	**UC-32.1 Filter by monthly revenue**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner filters revenue by selecting a month.		
Description:	This use case allows filtering reservation revenue by month.		
Preconditions:	Reservation revenue data exists.		
Postconditions:	Revenue data for the selected month is shown.		
Normal flow:	**32.1 Filter monthly** 1. Owner selects the month. 2. System filters and shows results.		
Alternative Flows:	N/A		
Exceptions:	**32.1.E1 No revenue for selected month**		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: /api/revenue/reservation?month=...		
Assumptions:	Revenue exists for selected months.		

2.3.73 Filter by weekly revenue of reservations

UC ID and Name:	**UC-32.2 Filter by weekly revenue**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner filters revenue by selecting a week.		
Description:	This use case allows filtering reservation revenue by week.		
Preconditions:	Reservation revenue data exists.		
Postconditions:	Revenue data for the selected week is shown.		
Normal flow:	**32.1 Filter weekly** 1. Owner selects the week. 2. System filters and shows results.		
Alternative Flows:	N/A		
Exceptions:	**32.1.E1 No revenue for selected week**		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: /api/revenue/reservation?week=...		
Assumptions:	Revenue exists for selected weeks.		

2.3.74 Filter by daily revenue of reservations

UC ID and Name:	**UC-32.3 Filter by daily revenue**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner filters revenue by selecting a day.		
Description:	This use case allows filtering reservation revenue by day.		
Preconditions:	Reservation revenue data exists.		
Postconditions:	Revenue data for the selected day is shown.		
Normal flow:	**32.3 Filter daily** 1. Owner selects the day. 2. System filters and shows results.		
Alternative Flows:	N/A		
Exceptions:	**32.3.E1 No revenue for selected day**		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: /api/revenue/reservation?day=...		
Assumptions:	Revenue exists for selected days.		

2.3.75 View revenue by dishes

UC ID and Name:	**UC-33 View revenue by dishes**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner clicks “Doanh thu từ món ăn” menu item in manage page.		
Description:	This use case allows restaurant owners to view total revenue generated from dishes.		
Preconditions:	Owner is logged in.		
Postconditions:	A revenue chart or list by dishes is shown.		
Normal flow:	**33.0 View dishes revenue** 1. Owner opens the revenue tab. 2. System loads total dishes revenue. 3. Displays amount and summary.		
Alternative Flows:	N/A		
Exceptions:	**33.0.E1 No data** Inform: “Không có dữ liệu doanh thu từ đặt món ăn.”		
Priority:	High		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: /api/revenue/food		
Assumptions:	Dishes have valid payment completed.		

2.3.76 Filter by monthly revenue of dishes

UC ID and Name:	**UC-33.1 Filter by monthly revenue**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner filters revenue by selecting a month.		
Description:	This use case allows filtering dishes revenue by month.		
Preconditions:	Dishes revenue data exists.		
Postconditions:	Revenue data for the selected month is shown.		
Normal flow:	**33.1 Filter monthly** 1. Owner selects the month. 2. System filters and shows results.		
Alternative Flows:	N/A		
Exceptions:	**33.1.E1 No revenue for selected month**		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: /api/revenue/food?month=...		
Assumptions:	Revenue exists for selected months.		

2.3.77 Filter by weekly revenue of reservations

UC ID and Name:	**UC-33.2 Filter by weekly revenue**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner filters revenue by selecting a week.		
Description:	This use case allows filtering dishes revenue by week.		
Preconditions:	Dishes revenue data exists.		
Postconditions:	Revenue data for the selected week is shown.		
Normal flow:	**33.1 Filter weekly** 1. Owner selects the week. 2. System filters and shows results.		
Alternative Flows:	N/A		
Exceptions:	**33.1.E1 No revenue for selected week**		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: /api/revenue/food?week=...		
Assumptions:	Revenue exists for selected weeks.		

2.3.78 Filter by daily revenue of reservations

UC ID and Name:	**UC-33.3 Filter by daily revenue**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	Owner filters revenue by selecting a day.		
Description:	This use case allows filtering dishes revenue by day.		
Preconditions:	Dishes revenue data exists.		
Postconditions:	Revenue data for the selected day is shown.		
Normal flow:	**33.3 Filter daily** 1. Owner selects the day. 2. System filters and shows results.		
Alternative Flows:	N/A		
Exceptions:	**33.3.E1 No revenue for selected day**		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: /api/revenue/food?day=...		
Assumptions:	Revenue exists for selected days.		

2.3.79 Chat with customer

UC ID and Name:	**UC-34 Chat with customer**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Restaurant Owner	Secondary Actors:	None
Trigger:	The owner clicks “Tin nhắn khách hàng” in restaurant owner page.		
Description:	This use case allows owners to communicate with customers via a chat interface.		
Preconditions:	Owner and customer have a related booking and followed		
Postconditions:	Message is sent and received with real time.		
Normal flow:	**34.0 Chat** 1. Owner opens a chat. 2. Owner choose specific customer. 2. Send a message. 3. System delivers it to the customer.		
Alternative Flows:	N/A		
Exceptions:	**34.0.E1 Connection lost** Inform: “Không thể gửi tin nhắn.”		
Priority:	High		
Frequency of Use:	Frequently		
Business Rules:	BR-55		
Other Information:	Endpoint: /api/chat/send		
Assumptions:	Both sides are online or able to receive messages.		

2.3.80 Logout in admin page

UC ID and Name:	**UC-35 Logout**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks “Đăng xuất” button in profile menu.		
Description:	This use case allows users to securely log out from the application.		
Preconditions:	Admin is logged in and authenticated.		
Postconditions:	Admin session is destroyed, and the system returns to the home or login screen.		
Normal flow:	**35.0 Logout** 1. Adminclicks logout. 2. System clears session and redirects.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	High		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: /api/auth/logout		
Assumptions:	Admin intends to leave the system securely.		

2.3.81 View list of categories

UC ID and Name:	**UC-36.1 View list of categories**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to view the list of categories.		
Description:	This use case allows the admin to view all existing categories.		
Preconditions:	Admin is logged in.		
Postconditions:	The list of categories is displayed.		
Normal flow:	**36.1 View list of categories** 1. Actor selects “View categories” option. 2. System retrieves category data. 3. System displays the list of categories.		
Alternative Flows:	N/A		
Exceptions:	**36.1.E1 Data not found** Inform the actor.		
Priority:	Medium		
Frequency of Use:	Often		
Business Rules:	N/A		
Other Information:	API endpoint: GET /api/categories		
Assumptions:	Data is up-to-date and accessible.		

2.3.82 Create category

UC ID and Name:	**UC-36.2 Create category**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to create a new category.		
Description:	This use case allows the admin to create a new category in the system.		
Preconditions:	Admin is logged in.		
Postconditions:	A new category is added to the system.		
Normal flow:	**36.2 Create category** 1. Actor selects “Create category” option. 2. System displays category creation form. 3. Actor enters category information and submits. 4. System saves the new category and confirms success.		
Alternative Flows:	N/A		
Exceptions:	**36.2.E1 Validation error** Show error messages.		
Priority:	Medium		
Frequency of Use:	Often		
Business Rules:	N/A		
Other Information:	API endpoint: POST /api/categories		
Assumptions:	All required data fields are provided.		

2.3.83 Modify category

UC ID and Name:	**UC-36.3 Modify category**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to edit a category.		
Description:	This use case allows the admin to update the detail of an existing category.		
Preconditions:	Admin is logged in.		
Postconditions:	Category information is updated in the system.		
Normal flow:	**36.3 Modify category** 1. Actor selects a category to edit. 2. System displays the category edit form. 3. Actor modifies category detail and submits. 4. System saves the changes and confirms success.		
Alternative Flows:	N/A		
Exceptions:	**36.3.E1 Validation error** Show error messages.		
Priority:	Medium		
Frequency of Use:	Often		
Business Rules:	N/A		
Other Information:	API endpoint: PUT /api/categories/{id}		
Assumptions:	All updates comply with data validation rules.		

2.3.84 Delete category

UC ID and Name:	**UC-36.4 Delete category**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to delete a category.		
Description:	This use case allows the admin to delete an existing category from the system.		
Preconditions:	Admin is logged in.		
Postconditions:	Category is removed from the system.		
Normal flow:	**36.4 Delete category** 1. Actor selects a category to delete. 2. System asks for confirmation. 3. Actor confirms deletion. 4. System removes the category and confirms success.		
Alternative Flows:	N/A		
Exceptions:	**36.4.E1 Category in use** Prevent deletion and inform the actor.		
Priority:	Medium		
Frequency of Use:	Often		
Business Rules:	N/A		
Other Information:	API endpoint: DELETE /api/categories/{id}		
Assumptions:	The category is not currently linked to active data.		

2.3.85 View list of restaurants

UC ID and Name:	**UC-37.1 View list of restaurants**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to view the list of restaurants.		
Description:	This use case allows the admin to view all restaurants in the system.		
Preconditions:	Admin is logged in.		
Postconditions:	The list of restaurants is displayed.		
Normal flow:	**37.1 View list of restaurants** 1. Actor opens the restaurant management page. 2. System loads restaurant data (name, email, phone, address, discount, rating, category, status). 3. Actor views the list; paging and search are available.		
Alternative Flows:	N/A		
Exceptions:	**37.1.E1 No data found** Inform: "Chưa có nhà hàng nào."		
Priority:	Medium		
Frequency of Use:	Often		
Business Rules:	N/A		
Other Information:	API endpoint: GET /api/restaurants		
Assumptions:	Data is accessible and up to date.		

2.3.86 Create restaurant

UC ID and Name:	**UC-37.2 Create restaurant**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks "Thêm Nhà Hàng" button on manage restaurant.		
Description:	This use case allows the admin to create a new restaurant.		
Preconditions:	Admin is logged in.		
Postconditions:	A new restaurant is created and appears in the list.		
Normal flow:	**37.2 Create restaurants** 1. Actor clicks "Thêm Nhà Hàng" button. 2. System displays the create form. 3. Actor enters required information and submits. 4. System validates inputs. 5. System creates the restaurant record. 6. System shows success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**37.2.E1 Validation failed** Show field errors. **37.2.E2 Duplicate email/phone** Inform the actor.		
Priority:	High		
Frequency of Use:	Sometimes		
Business Rules:	BR-59		
Other Information:	API endpoint: POST /api/restaurants		
Assumptions:	Required fields provided; unique constraints enforced.		

2.3.87 Modify restaurant

UC ID and Name:	**UC-37.3 Modify restaurant**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks "Sửa" button on a restaurant row.		
Description:	This use case allows the admin to update information of an existing restaurant.		
Preconditions:	Admin is logged in; target restaurant exists.		
Postconditions:	Restaurant information is updated.		
Normal flow:	**37.3 Modify restaurant** 1. Actor clicks "Sửa" button on a specific restaurant. 2. System shows current restaurant detail. 3. Actor edits fields and submits. 4. System validates inputs. 5. System updates the restaurant in the database. 6. System shows success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**37.3.E1 Restaurant not found** Inform and return to list. **37.3.E2 Validation failed** Show field errors. **37.3.E3 Duplicate email/phone** Inform the actor.		
Priority:	High		
Frequency of Use:	Sometimes		
Business Rules:	BR-48		
Other Information:	API endpoint: PUT /api/restaurants/{id}		

UC ID and Name:	**UC-37.3 Modify restaurant**
Assumptions:	Actor has edit permission.

2.3.88 Delete restaurant

UC ID and Name:	**UC-37.4 Delete restaurant**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to delete a restaurant.		
Description:	This use case allows the admin to delete an existing restaurant.		
Preconditions:	Admin is logged in; restaurant exists.		
Postconditions:	Restaurant is removed from the system.		
Normal flow:	**37.4 Delete restaurant** 1. Actor selects a restaurant to delete. 2. System asks for confirmation. 3. Actor confirms deletion. 4. System deletes the restaurant record. 5. System shows success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**37.4.E1 Restaurant in use (active orders/contracts)** Prevent deletion and inform the actor. **37.4.E2 Restaurant not found** Inform and refresh list.		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-48, BR-60		
Other Information:	API endpoint: DELETE /api/restaurants/{id}		

UC ID and Name:	**UC-37.4 Delete restaurant**
Assumptions:	Referential integrity is enforced; soft-delete may be applied if needed.

2.3.89 Modify status license

UC ID and Name:	**UC-38.1 Modify status license**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin modifies the status of a license request (Approve or Reject).		
Description:	This use case allows admins to approve or reject restaurant registration requests.		
Preconditions:	License request is pending. Admin is logged into the system.		
Postconditions:	If approved → restaurant license is active. If rejected → license status updated to "Rejected", the restaurant owner is notified with a rejection reason.		
Normal flow:	**38.1.0 Approve license** 1. Admin reviews the submitted license info. 2. Admin clicks Approve. 3. System updates the license status to Active. 4. Restaurant becomes live in the system.		
Alternative Flows:	**38.1.1 Reject license** 1. Admin reviews the submitted license info. 2. Admin inputs the rejection reason. 3. Admin clicks Reject. 4. System updates the license status to Rejected. 5. The restaurant owner is notified of the rejection.		
Exceptions:	**38.1.0.E1 Already approved** Inform: "Yêu cầu đã được duyệt." **38.1.1.E2 Already rejected** Inform: "Yêu cầu đã bị từ chối."		

UC ID and Name:	**UC-38.1 Modify status license**
Priority:	High
Frequency of Use:	Frequently
Business Rules:	BR-48, BR-53, BR-57
Other Information:	Endpoint: /api/restaurant/modify-license
Assumptions:	Valid license submission is in place.

2.3.90 View list of licenses

UC ID and Name:	**UC-38.2 View list of licenses**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to view the list of licenses.		
Description:	This use case allows the admin to view all licenses in the system.		
Preconditions:	Admin is logged in.		
Postconditions:	The list of licenses is displayed.		
Normal flow:	**38.2 View list of licenses** 1. Actor opens the restaurant owner management page. 2. System loads licenses. 3. Actor views the list; paging and search are available.		
Alternative Flows:	N/A		
Exceptions:	**38.2.E1 No data found** Inform: "Chưa có giấy phép kinh doanh nào."		
Priority:	High		
Frequency of Use:	Often		

UC ID and Name:	**UC-38.2 View list of licenses**
Business Rules:	N/A
Other Information:	API endpoint: GET /api/restaurants/license
Assumptions:	Data is accessible and up to date.

2.3.91 View list of customers

UC ID and Name:	**UC-39.1 View list of customer**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to view the list of customers.		
Description:	This use case allows the admin to view all customers in the system.		
Preconditions:	Admin is logged in.		
Postconditions:	The list of customers is displayed.		
Normal flow:	**39.1 View list of customers** 1. Actor opens the customer management page. 2. System loads customer data 3. Actor views the list; paging and search are available.		
Alternative Flows:	N/A		
Exceptions:	**39.1.E1 No data found** Inform: "Chưa có khách hàng nào."		

UC ID and Name:	**UC-39.1 View list of customer**
Priority:	Medium
Frequency of Use:	Often
Business Rules:	N/A
Other Information:	API endpoint: GET /api/customers
Assumptions:	Data is accessible and up to date.

2.3.92 Create customer

UC ID and Name:	**UC-39.2 Create customer**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks "Thêm Khách Hàng" button in manage customer page.		
Description:	This use case allows the admin to create a new customer.		
Preconditions:	Admin is logged in.		
Postconditions:	A new customer is created and appears in the list.		
Normal flow:	**39.2 Create customer** 1. Actor clicks "Thêm Khách Hàng" button. 2. System displays the create form. 3. Actor enters required information and submits. 4. System validates inputs. 5. System creates the customer record.		

UC ID and Name:	**UC-39.2 Create customer**
	6. System shows success message and refreshes the list.
Alternative Flows:	N/A
Exceptions:	**39.2.E1 Validation failed** Show field errors. **39.2.E2 Duplicate email/phone** Inform the actor.
Priority:	High
Frequency of Use:	Sometimes
Business Rules:	BR-48, BR-49, BR-50, BR-52, BR-59
Other Information:	API endpoint: POST /api/customers
Assumptions:	Required fields provided; unique constraints enforced.

2.3.93 Modify customer

UC ID and Name:	**UC-39.3 Modify customer**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks "Sửa" button on a customer row.		
Description:	This use case allows the admin to update information of an existing customer.		
Preconditions:	Admin is logged in; target customer exists.		
Postconditions:	Customer information is updated.		
Normal flow:	**39.3 Modify customer** 1. Admin clicks "Sửa" button on a specific customer.		

UC ID and Name:	**UC-39.3 Modify customer**
	2. The system shows current customer detail, including status (Active/Blocked). 3. Admin edits fields and optionally changes status. 4. Admin submits the form. 5. The system validates inputs. 6. System checks for duplicate email/phone. 7. The system updates the customer in the database. 8.System shows success message and refreshes the list.
Alternative Flows:	N/A
Exceptions:	**39.3.E1 Customer not found** Inform and return to list. **39.3.E2 Validation failed** Show field errors. **39.3.E3 Duplicate email/phone** Inform the actor.
Priority:	High
Frequency of Use:	Sometimes
Business Rules:	BR-48, BR-49, BR-50, BR-52, BR-59
Other Information:	API endpoint: PUT /api/customers/{id} Changing status to Blocked prevents customer from logging in.
Assumptions:	Actor has edit permission.

2.3.94 Delete customer

UC ID and Name:	**UC-39.4 Delete customer**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to delete a customer.		
Description:	This use case allows the admin to delete an existing customer.		
Preconditions:	Admin is logged in; customer exists.		

UC ID and Name:	**UC-39.4 Delete customer**
Postconditions:	The customer is removed from the system.
Normal flow:	**39.4 Delete customer** 1. The actor selects a customer to delete. 2. The system asks for confirmation. 3. Actor confirms deletion. 4. System deletes the customer record. 5. The system shows a success message and refreshes the list.
Alternative Flows:	N/A
Exceptions:	**39.4.E1 Customer not found** Inform and refresh list.
Priority:	Medium
Frequency of Use:	Sometimes
Business Rules:	BR-48
Other Information:	API endpoint: DELETE /api/customers/{id}
Assumptions:	Referential integrity is enforced; soft-delete may be applied if needed.

2.3.95 View list of voucher

UC ID and Name:	**UC-40.1 View list of voucher**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the “Danh sách voucher” menu item in the managed page.		

UC ID and Name:	**UC-40.1 View list of voucher**
Description:	This use case allows the admin to view all vouchers in the system.
Preconditions:	Admin is logged in.
Postconditions:	Vouchers are displayed.
Normal flow:	**40.1 View vouchers** 1. Admin opens the voucher section. 2. The system shows all vouchers.
Alternative Flows:	N/A
Exceptions:	**40.1.E1 No vouchers** Inform: “Hiện tại chưa có khuyến mãi nào.”
Priority:	Medium
Frequency of Use:	Regularly
Business Rules:	N/A
Other Information:	Endpoint: /api/voucher
Assumptions:	Vouchers exist.

2.3.96 Create voucher

UC ID and Name:	**UC-40.2 Create voucher**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025

UC ID and Name:	**UC-40.2 Create voucher**		
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the “Tạo voucher” button on the managed voucher page.		
Description:	This use case allows admin to create new vouchers system-wide.		
Preconditions:	Admin is logged in.		
Postconditions:	Voucher is saved and available.		
Normal flow:	**40.2 Create voucher** 1. Admin inputs info. 2. Clicks save. 3. System stores and publishes.		
Alternative Flows:	N/A		
Exceptions:	**40.2.E1 Invalid input** Inform: “Vui lòng nhập đúng dữ liệu.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-32, BR-33, BR-48		
Other Information:	Endpoint: /api/voucher/create		
Assumptions:	Voucher data is valid.		

2.3.97 Modify voucher

UC ID and Name:	**UC-40.3 Modify voucher**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks “Sửa” button on specified voucher.		
Description:	This use case allows admin to edit vouchers.		
Preconditions:	Vouchers exist.		
Postconditions:	Voucher is updated.		
Normal flow:	**40.3 Modify voucher** 1. Admin selects a voucher. 2. Click the edit button. 3. Fill in information and submit 4. System updates status.		
Alternative Flows:	N/A		
Exceptions:	**40.3.E1 Voucher not found** Inform and return to list. **40.3.E2 Validation failed** Show field errors.		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-32, BR-33, BR-48		
Other Information:	Endpoint: /api/voucher/update		
Assumptions:	Admin confirms update.		

2.3.98 Delete voucher

UC ID and Name:	**UC-40.4 Delete voucher**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the “Xóa” button on the specified voucher.		
Description:	This use case allows admin to delete or deactivate vouchers.		
Preconditions:	Vouchers exist.		
Postconditions:	Voucher is removed or hidden.		
Normal flow:	**40.4 Delete voucher** 1. Admin selects a voucher. 2. Clicks delete. 3. System updates status.		
Alternative Flows:	N/A		
Exceptions:	**40.4.E1 Voucher not found** Inform and return to list.		
Priority:	Medium		
Frequency of Use:	Regularly		
Business Rules:	BR-48		
Other Information:	Endpoint: /api/voucher/delete		
Assumptions:	Admin confirms deletion.		

2.3.99 View list of notifications

UC ID and Name:	**UC-41.1 View list of notifications**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks “Quản lý thông báo” menu item in manage page.		
Description:	This use case allows the admin to view all system notifications.		
Preconditions:	Admin is logged in.		
Postconditions:	Notification list is displayed.		
Normal flow:	**41.1 View notifications** 1. Admin accesses notification panel. 2. The system shows notifications.		
Alternative Flows:	N/A		
Exceptions:	**41.1.E1 No notifications** Inform: “Hiện tại chưa có thông báo nào.”		
Priority:	Medium		
Frequency of Use:	Regularly		
Business Rules:	N/A		
Other Information:	Endpoint: /api/notifications		
Assumptions:	Notifications exist.		

2.3.100 Create notification

UC ID and Name:	**UC-41.2 Create notification**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks “Thêm thông báo” button on the manage notification page.		
Description:	This use case allows the admin to create a new system-wide notification.		
Preconditions:	Admin is logged in.		
Postconditions:	Notification is sent to target users.		
Normal flow:	**41.2 Create notification** 1. Admin writes title, content and type. 2. Clicks submit button. 3. The system delivers.		
Alternative Flows:	N/A		
Exceptions:	**41.2.E1 Missing content** Inform: “Vui lòng nhập đúng định dạng.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-48, BR-56		
Other Information:	Endpoint: POST /api/notifications		
Assumptions:	Content is valid and targeted correctly.		

2.3.101 Modify notification

UC ID and Name:	**UC-41.3 Modify notification**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks edit button at a notification.		
Description:	This use case allows the admin to delete or remove old notifications.		
Preconditions:	Notification exists.		
Postconditions:	Notification is deleted.		
Normal flow:	**41.3 Modify notification** 1. Admin clicks edit button . 2. Fill in information that needs to be updated. 3. System shows updated notification.		
Alternative Flows:	N/A		
Exceptions:	**41.3.E1 Notification not found** Inform and return to list. **41.3.E2 Notification failed** Show field errors.		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-48, BR-56		
Other Information:	Endpoint: PUT /api/notifications/{notificationId}		
Assumptions:	Notification is safe to remove.		

2.3.102 Delete notification

UC ID and Name:	**UC-41.4 Delete notification**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the delete button at a notification.		
Description:	This use case allows the admin to delete or remove old notifications.		
Preconditions:	Notification exists.		
Postconditions:	Notification is deleted.		
Normal flow:	**41.4 Delete notification** 1. Admin clicks delete. 2. The system removes it.		
Alternative Flows:	N/A		
Exceptions:	**41.4.E1 Notification not found** Inform and return to list.		
Priority:	Medium		
Frequency of Use:	Rarely		
Business Rules:	BR-48		
Other Information:	Endpoint: DELETE /api/notifications/{notificationID}		
Assumptions:	Notification is safe to remove.		

2.3.103 Log in as administrator

UC ID and Name:	**UC-42 Login with account**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	The user clicks the "Đăng nhập" button on the login page.		
Description:	A function to login and access to the administrator page.		
Preconditions:	The user must have a valid account.		
Postconditions:	The user is redirected to the managed page with appropriate permissions.		
Normal flow:	**42.0 Login by admin account.** 1. Guest access to the login page. 2. Guest enters username and password. 3. The user clicks the login button. 4. The system will redirect users to the managed page.		
Alternative Flows:	**N/A**		
Exceptions:	**42.0.E1 Guest enters username or password fields blank.** Inform users cannot leave the input field blank. **42.0.E2 Guest input the wrong username or password.** Inform users have input an invalid username or password.		
Priority:	High		
Frequency of Use:	Regularly		
Business Rules:	N/A		
Other Information:	Endpoint API: /api/admin/login Passwords need to be encrypted before being saved to the database.		
Assumptions:	An account must exist.		

2.3.104 View profile admin

UC ID and Name:	**UC-43.1 View profile**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Guest clicks on “Thông tin tài khoản” menu item from profile menu.		
Description:	This use case allows the admin to view information stored in the system.		
Preconditions:	The user is logged in.		
Postconditions:	Profile information is retrieved and displayed.		
Normal flow:	**43.1 View profile** 1. Admin accesses the profile section. 2. The system retrieves user data. 3. The system displays name, email, phone, avatar, birthday, etc…		
Alternative Flows:	N/A		
Exceptions:	N/A		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-48		
Other Information:	Endpoint: /api/admin/profile		
Assumptions:	Guest session is valid and the backend is available.		

2.3.105 Modify profile admin

UC ID and Name:	**UC-43.2 Modify profile**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	The admin changes information on the Profile form and clicks save changes.		
Description:	This use case allows admin to update their profile information (name, phone, avatar…).		
Preconditions:	Admin is logged in and viewing their own profile.		
Postconditions:	Profile information is updated in the system.		
Normal flow:	**43.2 Modify profile.** 1. Admin opens the Profile page. 2. Admin edits one or more fields (name, gender, phone, address, birthday). 3. Admin clicks the save button. 4. The system validates the input. 5. The system saves the changes. 6. The system shows a success message and displays updated information.		
Alternative Flows:	N/A		
Exceptions:	**43.2.E1 Invalid phone/email format** Inform: “Thông tin không hợp lệ.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-48		
Other Information:	Endpoint: /api/admin/profile/update		

UC ID and Name:	**UC-43.2 Modify profile**
Assumptions:	Admin provides valid data and has permission to update their profile.

2.3.106 View list of blogs

UC ID and Name:	**UC-44.1 View list of blogs**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Owner clicks “Quản lý bài viết” menu item from the managed page.		
Description:	This use case allows the admin to view all blogs posted by them.		
Preconditions:	Admin is logged in.		
Postconditions:	The system shows a list of created blogs.		
Normal flow:	**44.1 View blogs** 1. The admin clicks the blog tab. 2. The system loads blogs. 3. Displays list.		
Alternative Flows:	N/A		
Exceptions:	**44.1.E1 No blogs found** Inform: “Bạn chưa đăng blog nào.”		
Priority:	Medium		
Frequency of Use:	Regularly		
Business Rules:	View list of blogs		
Other Information:	Endpoint: GET /api/blog		

UC ID and Name:	**UC-44.1 View list of blogs**
Assumptions:	Blogs have been created by the admin.

2.3.107 Create blog

UC ID and Name:	**UC-44.2 Create blog**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks “Thêm bài viết” button on manage blog page.		
Description:	This use case allows the admin to create a new blog post.		
Preconditions:	Admin is logged in.		
Postconditions:	The blog is saved and appears in the list.		
Normal flow:	**44.2 Create blog** 1. The owner enters the title, content, and image. 2. Clicks publish. 3. Blog is saved.		
Alternative Flows:	N/A		
Exceptions:	**44.2.E1 Missing fields** Inform: “Vui lòng điền tiêu đề và nội dung.”		
Priority:	Medium		
Frequency of Use:	Sometimes		

UC ID and Name:	**UC-44.2 Create blog**
Business Rules:	BR-48, BR-59
Other Information:	Endpoint: POST /api/blog
Assumptions:	Admin writes valid content.

2.3.108 Modify blog

UC ID and Name:	**UC-44.3 Modify blog**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks edit button in a blog page..		
Description:	This use case allows the admin to update a previously posted blog.		
Preconditions:	The blog exists and belongs to the admin.		
Postconditions:	The blog is updated with new content.		
Normal flow:	**44.3 Modify blog** 1. Admin selects a blog. 2. Edits content. 3. Click the update button. 4. The system saves changes.		
Alternative Flows:	N/A		
Exceptions:	**44.3.E1 Invalid input** Inform: “Không thể lưu bài viết.”		

UC ID and Name:	**UC-44.3 Modify blog**
Priority:	Medium
Frequency of Use:	Sometimes
Business Rules:	BR-48, BR-59
Other Information:	Endpoint: PUT /api/blog/{blogId}
Assumptions:	Blog belongs to the admin and is not deleted.

2.3.109 Delete blog

UC ID and Name:	**UC-44.4 Delete blog**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the “Xóa” button in a specified blog.		
Description:	This use case allows the admin to delete a blog post.		
Preconditions:	The blog exists and belongs to the admin.		
Postconditions:	The blog is removed or marked inactive.		
Normal flow:	**44.4 Delete blog** 1. Admin clicks the delete button. 2. Confirms delete. 3. The system removes blogs.		
Alternative Flows:	N/A		

UC ID and Name:	**UC-44.4 Delete blog**
Exceptions:	**44.4.E1 Blog not found or locked** Inform: “Không thể xóa blog.”
Priority:	Medium
Frequency of Use:	Sometimes
Business Rules:	BR-48
Other Information:	Endpoint: DELETE /api/blog/{blogId}
Assumptions:	Admin confirms deletion.

2.3.110 View list of orders

UC ID and Name:	**UC-45.1 View list of orders**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin access to "Đơn đặt món ăn" menu item in management page.		
Description:	This feature allows the Admin to view all orders in the system.		
Preconditions:	Admin is logged into the system. Orders exist in the database.		
Postconditions:	The list of orders is displayed to the Admin.		
Normal flow:	**45.1 View list of orders** 1. Admin selects the "Manage Orders" menu. 2.The system retrieves all orders from the database.		

UC ID and Name:	**UC-45.1 View list of orders**
	3.The system displays the list of orders.
Alternative Flows:	N/A
Exceptions:	**45.1.E1 No orders found** Inform: "Không có đơn hàng nào."
Priority:	Medium
Frequency of Use:	Often
Business Rules:	N/A
Other Information:	Endpoint: GET /api/orders
Assumptions:	Orders are correctly stored and retrievable from the database.

2.3.111 Modify status order

UC ID and Name:	**UC-45.2 Modify status order**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to update an order from the order list		
Description:	This feature allows the Admin to update the status of an existing order in the system.		
Preconditions:	Admin is logged into the system. Orders exist in the database.		

UC ID and Name:	**UC-45.2 Modify status order**
Postconditions:	Order status is updated successfully.
Normal flow:	**45.2 Update order** 1. The system displays the order status. 2. Admin select status of an order. 3. Admin clicks "Cập nhật" button. 4. The system updates the order in the database and shows a success message.
Alternative Flows:	N/A
Exceptions:	**45.2.E1 Order not found** Inform the Admin: "Đơn hàng không tồn tại."
Priority:	Medium
Frequency of Use:	Often
Business Rules:	BR-26, BR-48
Other Information:	Endpoint: PUT /api/orders/{id}
Assumptions:	Admin has the necessary permissions to update orders.

2.3.112 View list of reservations

UC ID and Name:	**UC-46.1 View list of reservations**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin access to "Đơn đặt bàn" menu item in management page.		

UC ID and Name:	**UC-46.1 View list of reservations**
Description:	This feature allows the Admin to view all reservations in the system.
Preconditions:	Admin is logged into the system. Reservations exist in the database.
Postconditions:	The list of reservations is displayed to the Admin.
Normal flow:	**46.1 View reservations** 1. Admin selects the managed reservations menu. 2. The system retrieves all reservations from the database. 3. The system displays the list of reservations.
Alternative Flows:	N/A
Exceptions:	**46.1.E1 No reservations found** Inform: "Không có đơn đặt bàn nào."
Priority:	Medium
Frequency of Use:	Often
Business Rules:	N/A
Other Information:	Endpoint: GET /api/reservations
Assumptions:	Reservations are correctly stored and retrievable from the database.

2.3.113 Modify status reservation

UC ID and Name:	**UC-46.2 Modify status reservation**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025

UC ID and Name:	**UC-46.2 Modify status reservation**		
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to update a reservation from the reservation list.		
Description:	This feature allows the Admin to update the status of an existing reservation in the system.		
Preconditions:	Admin is logged into the system. Reservations exist in the database.		
Postconditions:	Reservations status is updated successfully.		
Normal flow:	**46.2 Update reservations** 1. The system displays the reservation status. 2. Admin select status of a reservation . 3. Admin clicks "Cập nhật" button. 4. The system updates the reservation in the database and shows a success message.		
Alternative Flows:	N/A		
Exceptions:	**46.2.E1 Order not found** Inform the Admin: "Đơn hàng không tồn tại."		
Priority:	Medium		
Frequency of Use:	Often		
Business Rules:	BR-48		
Other Information:	Endpoint: PUT /api/reservations/{id}		
Assumptions:	Admin has the necessary permissions to update reservations.		

2.3.114 View list of restaurant owners

UC ID and Name:	**UC-47.1 View list of restaurant owners**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to view the list of restaurant owners.		
Description:	This use case allows the admin to view all restaurant owners in the system.		
Preconditions:	Admin is logged in.		
Postconditions:	The list of restaurant owners is displayed.		
Normal flow:	**47.1 View list of restaurant owners** 1. The actor opens the restaurant owner's management page. 2. The system loads restaurant owners. 3. Actors view the list; paging and search are available.		
Alternative Flows:	N/A		
Exceptions:	**47.1.E1 No data found** Inform: "Chưa có chủ nhà hàng nào."		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	API endpoint: GET /api/restaurantOwners		
Assumptions:	Data is accessible and up to date.		

2.3.115 Create restaurant owner

UC ID and Name:	**UC-47.2 Create restaurant owner**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks "Thêm chủ nhà hàng" button in manage restaurant owners page.		
Description:	This use case allows the admin to create a new restaurant owner.		
Preconditions:	Admin is logged in.		
Postconditions:	A new restaurant owner is created and appears in the list.		
Normal flow:	**47.2 Create restaurant owner** 1. Actor clicks "Thêm Chủ Nhà Hàng" button. 2. The system displays the create form. 3. The actor enters required information and submits. 4. The system validates inputs. 5. The system creates the restaurant owner record. 6. The system shows a success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**47.2.E1 Validation failed** Show field errors. **47.2.E2 Duplicate email/phone** Inform the actor.		
Priority:	High		
Frequency of Use:	Sometimes		
Business Rules:	BR-48		
Other Information:	API endpoint: POST /api/restaurantOwners		
Assumptions:	Required fields provided; unique constraints enforced.		

2.3.116 Modify restaurant owner

UC ID and Name:	**UC-47.3 Modify restaurant owner**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks edit button on a restaurant owner row.		
Description:	This use case allows the admin to update information of an existing restaurant owner.		
Preconditions:	Admin is logged in; target restaurant owner exists.		
Postconditions:	Restaurant owner information is updated.		
Normal flow:	**47.3 Modify restaurant owner** 1. Actor clicks the edit button on a specific restaurant owner. 2. System shows current restaurant owner detail. 3. Actor edits fields and submits. 4. System validates inputs. 5. System updates the restaurant owner in the database. 6. System shows a success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**47.3.E1 Restaurant owner not found** Inform and return to list. **47.3.E2 Validation failed** Show field errors. **47.3.E3 Duplicate email/phone** Inform the actor.		
Priority:	High		
Frequency of Use:	Sometimes		
Business Rules:	BR-48		
Other Information:	API endpoint: PUT /api/restaurantOwners/{id}		
Assumptions:	The actor has edit permission.		

2.3.117 Delete restaurant owner

UC ID and Name:	**UC-47.4 Delete restaurant owner**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to delete a restaurant owner.		
Description:	This use case allows the admin to delete an existing restaurant owner.		
Preconditions:	Admin is logged in; restaurant exists.		
Postconditions:	The restaurant owner is removed from the system.		
Normal flow:	**47.4 Delete restaurant owner** 1. Actor selects a restaurant owner to delete. 2. System asks for confirmation. 3. Actor confirms deletion. 4. System deletes the restaurant owner record. 5. System shows success message and refreshes the list.		
Alternative Flows:	N/A		
Exceptions:	**47.4.E1 Restaurant owner in use (active orders/contracts)** Prevent deletion and inform the actor. **47.4.E2 Restaurant owner not found** Inform and refresh list.		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-48		
Other Information:	API endpoint: DELETE /api/restaurantOwner/{id}		
Assumptions:	Referential integrity is enforced; soft-delete may be applied if needed.		

2.3.118 View list of reports

UC ID and Name:	**UC-48.1 View list of reports**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks “Báo cáo từ người dùng” menu item from the managed page.		
Description:	This use case allows the admin to view all reports by the customer.		
Preconditions:	Admin is logged in.		
Postconditions:	The system shows a list of created reports.		
Normal flow:	**48.1 View reports** 1. Admin clicks the report tab. 2. System loads reports. 3. Displays list.		
Alternative Flows:	N/A		
Exceptions:	**48.1.E1 No reports found** Inform: “Chưa có đơn báo cáo nào.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/report		
Assumptions:	Reports have been created by the customer.		

2.3.119 Response report

UC ID and Name:	**UC-48.2 Response report**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the response button in a specified report.		
Description:	This use case allows the admin to respond to a specified report of a customer.		
Preconditions:	Report exists.		
Postconditions:	The report is responded with content.		
Normal flow:	**48.2 Response report** 1. Admin selects a specific report. 2. Enter content response. 3. Clicks confirm response. 4. The system sends responses to the customer.		
Alternative Flows:	N/A		
Exceptions:	**48.1.E1 No reports found** Inform: “Báo cáo không tồn tại.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-48		
Other Information:	Endpoint: POST /api/report/response/{reportId}		
Assumptions:	The report belongs to the admin and is not deleted.		

2.3.120 Delete report

UC ID and Name:	**UC-48.3 Delete report**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the delete button in a specified report.		
Description:	This use case allows the admin to delete a specified report.		
Preconditions:	The report exists and belongs to the admin.		
Postconditions:	The report is removed or marked inactive.		
Normal flow:	**48.3 Delete report** 1. Admin clicks the delete button. 2. Confirms delete. 3. The system removes reports.		
Alternative Flows:	N/A		
Exceptions:	**48.3.E1 Report not found or locked** Inform: “Không thể xóa bài báo cáo này.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-48		
Other Information:	Endpoint: DELETE /api/report/{blogId}		
Assumptions:	Admin confirms deletion.		

2.3.121 View list of contacts

UC ID and Name:	**UC-49.1 View list of contacts**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks “Liên hệ từ khách hàng” menu item from the managed page.		
Description:	This use case allows the admin to view all contacts by the user.		
Preconditions:	Admin is logged in.		
Postconditions:	The system shows a list of created contacts.		
Normal flow:	**49.1 View contacts** 1. Admin clicks the contacts tab. 2. The system loads contacts . 3. Displays list.		
Alternative Flows:	N/A		
Exceptions:	**49.1.E1 No contacts found** Inform: “Chưa có đơn liên hệ nào.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/contact		
Assumptions:	Contacts have been created by the user.		

2.3.122 Response contact

UC ID and Name:	**UC-49.2 Response contact**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the response button in a specified contact.		
Description:	This use case allows the admin to respond to a specified contact of a user.		
Preconditions:	Contact exists.		
Postconditions:	The contact is responded with content.		
Normal flow:	**49.2 Response contact** 1. Admin selects a specific contact. 2. Enter content response. 3. Clicks confirm button. 4. The system sends responses to the user.		
Alternative Flows:	N/A		
Exceptions:	**49.1.E1 No contacts found** Inform: “Không tìm thấy đơn liên hệ.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-48		
Other Information:	Endpoint: POST /api/contact/response/{reportId}		
Assumptions:	The contact belongs to the admin and is not deleted.		

2.3.123 Delete contact

UC ID and Name:	**UC-49.3 Delete contact**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the delete button in a specified contact.		
Description:	This use case allows the admin to delete a specified contact.		
Preconditions:	The contact exists and belongs to the admin.		
Postconditions:	The contact is removed or marked inactive.		
Normal flow:	**49.3 Delete contact** 1. Admin clicks delete. 2. Click the confirm button. 3. The system removes contacts.		
Alternative Flows:	N/A		
Exceptions:	**49.3.E1 Contact not found or locked** Inform: “Không thể xóa đơn liên hệ này.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	BR-48		
Other Information:	Endpoint: DELETE /api/contact/{blogId}		
Assumptions:	Admin confirms deletion.		

2.3.124 View list of payments

UC ID and Name:	**UC-50.1 View list of payments**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks “Quản lý đơn thanh toán” menu item from the managed page.		
Description:	This use case allows the admin to view all payments by the customer.		
Preconditions:	Admin is logged in.		
Postconditions:	The system shows a list of created payments.		
Normal flow:	**50.1 View list of payments** 1. The owner clicks the payments tab. 2. The system loads payments . 3. Displays list.		
Alternative Flows:	N/A		
Exceptions:	**50.1.E1 No payments found** Inform: “Chưa có đơn hàng nào đã thanh toán.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/payment		
Assumptions:	Payments have been created by the user.		

2.3.125 Print payment

UC ID and Name:	**UC-50.2 Print payment**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the print button on a specific payment.		
Description:	This use case allows the admin to print specific payment.		
Preconditions:	Admin is logged in. Payment exists.		
Postconditions:	The system prints payment.		
Normal flow:	**50.2 Print payment** 1. The owner clicks the payments tab. 2. The system loads payments. 3. The owner clicks on the print button. 4. The printer will print that payment.		
Alternative Flows:	N/A		
Exceptions:	**50.2.E1 No payments found** Inform: “Chưa có đơn hàng nào đã thanh toán.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/payment/print		
Assumptions:	Payments have been printed by the printer.		

2.3.126 Dashboard in admin page

UC ID and Name:	**UC-51 View dashboard**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin clicks the “Thống kê” menu item from the managed page.		
Description:	This use case allows the administrator to view all sales statistics on table reservations and food orders of the restaurants.		
Preconditions:	Admin is logged in.		
Postconditions:	The system shows a dashboard of restaurants.		
Normal flow:	**51.0 View dashboard** 1. The admin clicks the dashboard tab. 2. The system loads all sales statistics on table reservations and food orders of the restaurants. 3. Displays list.		
Alternative Flows:	N/A		
Exceptions:	**51.1.E1 No data found** Inform: “Chưa có dữ liệu.”		
Priority:	High		
Frequency of Use:	Regularly		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/payment/{restaurantID} GET /api/reservation/{restaurantID} GET /api/order/{restaurantID}		
Assumptions:	Restaurants must have revenue from table reservations and food orders. Sales need to be updated at a specific time.		

2.3.127 View system logs

UC ID and Name:	**UC-52 View system logs**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin wants to view the history of actions performed by admins.		
Description:	This use case allows the system to automatically record all actions performed by admins (such as create, update, delete on customers, restaurants, menus, reservations, etc.) and allows an admin to view these logs.		
Preconditions:	Admin is logged in. System logging is enabled.		
Postconditions:	Admin activity is recorded in the log table. Admin can view a list of logged actions.		
Normal flow:	**52.0 View system logs** 1. Actor performs an action in the system (e.g., create/update/delete a customer, restaurant, menu, or reservation). 2. System automatically records the following detail in the admin log: 3. Actor navigates to the "Admin Log" page. 4. System retrieves logs from the database and displays them in a table with columns: Timestamp, Admin, Action, Target, Detail. 5. Actor can filter logs by date, admin, action type, or target entity. 6. Actor can optionally export the log as CSV/PDF.		
Alternative Flows:	N/A		
Exceptions:	**52.0.E1 No logs available** System displays a message "No actions recorded yet." **52.0.E2 Database unavailable** Show error message and retry option.		
Priority:	High		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	API endpoint (optional): GET /api/admin/logs		
Assumptions:	Both sides are online or able to receive messages.		

2.3.128 Checkout order

UC ID and Name:	**UC-53 Checkout order**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Triggered when the customer clicks the "Mua Hàng" button on the cart page.		
Description:	Allows the customer to confirm and place an order for the items currently in their cart, specifying delivery information and applying vouchers if available.		
Preconditions:	Customer is logged in and has at least one item selected in the cart.		
Postconditions:	The order is created in the system, and its status is set to "Pending Confirmation" or equivalent.		
Normal flow:	**53.0 Checkout order** 1. Customer clicks "Mua Hàng" button on the cart page. 2. System navigates to the Checkout page, displaying delivery information form and order detail. 3. Customer fills in required delivery information. 4. Customer can enter or select a voucher code (optional). 5. Customer clicks "Xác nhận đơn hàng". 6. System validates input data and voucher (if provided). 7. System creates the order, saves it in the database, and redirects to the Delivery Tracking page showing order status.		
Alternative Flows:	N/A		
Exceptions:	**53.0.E1 System error during order creation** The system displays an error message and keeps the customer on the checkout page for retry.		
Priority:	High		
Frequency of Use:	Frequently		
Business Rules:	BR-19, BR-27, BR-28		
Other Information:	Endpoint: /api/order/checkout		
Assumptions:	The customer has a stable internet connection during the checkout process.		

2.3.129 View list of notifications

UC ID and Name:	**UC-54 View list of notifications**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The customer clicks the notification icon in the header.		
Description:	This use case allows the customer to view all notifications by system.		
Preconditions:	The customer is logged in and notification exists.		
Postconditions:	The system shows a list of notifications.		
Normal flow:	**54.0 View list of notifications** 1. The owner clicks the notification icon. 2. The system loads notifications. 3. Displays list.		
Alternative Flows:	N/A		
Exceptions:	**54.0.E1 No notifications found** Inform: “Chưa có thông báo nào.”		
Priority:	Medium		
Frequency of Use:	Regularly		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/notification		
Assumptions:	Notifications have been created by system.		

2.3.130 View list of comments

UC ID and Name:	**UC-55.1 View list of comments**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer click comment input in the post detail.		
Description:	This use case allows users to view a list of comments.		
Preconditions:	The customer is logged in.		
Postconditions:	The list of comments created by the customer is displayed.		
Normal flow:	**55.1 View comments** 1. Guest accesses the post detail. 2. The system retrieves all comments created from the post. 3. Comments are displayed.		
Alternative Flows:	N/A		
Exceptions:	**55.1.E1 No comments available** Inform: “Chưa có comment nào.”		
Priority:	Medium		
Frequency of Use:	Often		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/comments		
Assumptions:	Guest has at least one comment.		

2.3.131 Create comment

UC ID and Name:	**UC-55.2 Create comment**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer comment in the post detail.		
Description:	Allows the customer to create comments in post.		
Preconditions:	Customer is logged in.		
Postconditions:	The comment is saved in the system and displayed on the customer's timeline.		
Normal flow:	**55.2 Create comment** 1. Customer opens the post detail. 2. Customer enters text content and uses emoji. 3. Customer clicks the Publish button. 4. System saves the comment. 5. System displays the new comment on the timeline.		
Alternative Flows:	N/A		
Exceptions:	**55.2.E1 Empty content:** Show error requiring text.		
Priority:	Medium		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: POST /api/comments.		
Assumptions:	Guest provides valid input.		

2.3.132 Modify comment

UC ID and Name:	**UC-55.3 Modify comment**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer clicks “Chỉnh sửa” button in specific comment.		
Description:	This use case allows users to edit previously created comments.		
Preconditions:	The customer is logged in and owns the comment.		
Postconditions:	The comment is updated with new content.		
Normal flow:	**55.3 Modify comment** 1. Guest clicks edit button. 2. Guests edit and save. 3. System updates comment.		
Alternative Flows:	N/A		
Exceptions:	**55.3.E1 Empty content:** Show error requiring text.		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: PUT /api/comment/{commentId}		
Assumptions:	The user is the author of the comment.		

2.3.133 Delete comment

UC ID and Name:	**UC-55.4 Delete comment**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The customer clicks the “Xóa” button in a comment.		
Description:	This use case allows users to remove their own comments from the system.		
Preconditions:	The customer is logged in and owns the comment.		
Postconditions:	The comment is permanently removed or marked as deleted.		
Normal flow:	**55.4 Delete comment** 1. Guest clicks the delete button. 2. The system asks for confirmation. 3. Guest confirms. 4. The system deletes comments.		
Alternative Flows:	N/A		
Exceptions:	**55.4.E1 Empty content:** Show error requiring text.		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: DELETE /api/posts/{postId}		
Assumptions:	The customer is the author of the comment.		

2.3.134 View list of interactions

UC ID and Name:	**UC-56.1 View list of interactions**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customers click the number of interactions beside the like icon in the post.		
Description:	This use case allows users to view a list of likes in the post.		
Preconditions:	The customer is logged in.		
Postconditions:	The list of likes created by the customer is displayed.		
Normal flow:	**56.1 View list of interactions** 1. Guest accesses the post. 2. The system retrieves all likes created from the post. 3. Likes are displayed.		
Alternative Flows:	N/A		
Exceptions:	**56.1.E1 No interactions** Inform: “Chưa có lượt tương tác nào.”		
Priority:	Medium		
Frequency of Use:	Often		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/interaction		
Assumptions:	Customer has at least one like.		

2.3.135 Create interaction

UC ID and Name:	**UC-56.2 Create interaction**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customer interaction in the post.		
Description:	Allows the customer to create like in post.		
Preconditions:	The user is logged in.		
Postconditions:	The like is saved in the system and displayed on the customer's timeline.		
Normal flow:	**56.2 Create interaction** 1. The customer opens the post page. 2. Customer select emoji. 3. System saves the interaction type. 4. The system displays the new like on the timeline.		
Alternative Flows:	N/A		
Exceptions:	**56.1.E1 No post** Inform: "Không tìm thấy bài đăng này."		
Priority:	Medium		
Frequency of Use:	Frequently		
Business Rules:	N/A		
Other Information:	Endpoint: POST /api/interaction.		
Assumptions:	Customer provides valid input.		

2.3.136 Modify interaction

UC ID and Name:	**UC-56.3 Modify interaction**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	Customers choose new emoji in post.		
Description:	This use case allows customers to edit previously created likes.		
Preconditions:	The customer is logged in and owns the likes.		
Postconditions:	The like is updated.		
Normal flow:	**56.3 Modify interaction** 1. Customers choose new emoji. 2. System updates like.		
Alternative Flows:	N/A		
Exceptions:	**56.3.E1 No post** Inform: “Không tìm thấy bài đăng này.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: PUT /api/interaction/{commentId}		
Assumptions:	The user is the author of the comment.		

2.3.137 Delete interaction

UC ID and Name:	**UC-56.4 Delete interaction**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The customer re-selects the selected emoji.		
Description:	This use case allows users to remove their own likes from the system.		
Preconditions:	The customer is logged in and owns the likes.		
Postconditions:	The like is permanently removed.		
Normal flow:	**56.4 Delete interaction** 1. The customer re-selects the selected emoji. 2. The system deletes likes.		
Alternative Flows:	N/A		
Exceptions:	**56.4.E1 No post** Inform: “Không tìm thấy bài đăng này.”		
Priority:	Medium		
Frequency of Use:	Sometimes		
Business Rules:	N/A		
Other Information:	Endpoint: DELETE /api/interaction/{postId}		
Assumptions:	Customer is the author of the like.		

2.3.138 *Read blog by voice*

UC ID and Name:	**UC-57 Read blog by voice**		
Create By:	Tran Minh Thang	Date Created:	08/08/2025
Primary actor:	Customer	Secondary Actors:	None
Trigger:	The user clicks the Start Voice button on the blog detail page.		
Description:	This use case allows the user to listen to a blog article through AI-generated speech. When the user clicks "Start Voice," the system retrieves the blog content, sends it to a TTS service to convert text to speech, and plays the generated audio directly in the browser.		
Preconditions:	The user is viewing a blog. The blog content exists and is in text format. The browser supports audio playback.		
Postconditions:	The blog content is read aloud in full to the user.		
Normal flow:	**57.0 Read blog by voice** 1. The customer opens a blog post detail page. 2. The customer clicks the **Start Voice** button. 3. The system retrieves the blog content. 4. The system sends the text content to the TTS service. 5. The TTS service returns an audio file. 6. The system plays the audio file in the user's browser.		
Alternative Flows:	N/A		
Exceptions:	**57.0.E1: TTS service unavailable** Display an error message to the user. **57.0.E2 Blog content is empty or unsupported** Disable voice reading. **57.0.Audio playback error** Display an error and allow retry.		

Priority:	Medium
Frequency of Use:	Sometimes
Business Rules:	BR-58
Other Information:	API endpoint to call the TTS service. Audio playback implemented via HTML5 <audio> tag or Web Speech API.
Assumptions:	Stable internet connection. The TTS service and the server are operation

2.3.139 Register

UC ID and Name:	**UC-58 Register**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	Customer	Secondary Actors:	Mail Service
Trigger:	The user clicks the "Register" button on the registration form after entering the required information.		
Description:	This use case allows a user to create a new account in the system. After entering registration information, the system validates the inputs, generates an OTP code, and sends it to the user's email via the Mail Service. The user must confirm the OTP to successfully activate the account.		
Preconditions:	The user does not already have an account in the system. Internet connection and mail service are available.		
Postconditions:	The account is successfully created once the user verifies the OTP sent to their email. If the OTP is invalid or expired, the account is not activated.		
Normal flow:	**58.0 Register** 1. The user fills in the registration form and clicks Register. 2. The system validates the input data. 3. The system generates an OTP and stores it temporarily. 4. The system sends the OTP to the user's email through the Mail Service. 5. The user enters the OTP in the verification form. 6. The system validates the OTP. 7. If successful, the system activates the account and redirects the user to the login page.		

UC ID and Name:	**UC-58 Register**
Alternative Flows:	N/A
Exceptions:	**58.0.E1 Invalid input:** The system displays an error message. **58.0.E2 Email not received:** The user can request the system to resend the OTP. **58.0.E3 Invalid or expired OTP:** The system notifies the user and requires a new attempt **58.0.E4 Mail Service unavailable:** The system informs the user that the email cannot be sent and asks to try again later.
Priority:	High
Frequency of Use:	Frequently
Business Rules:	BR-49, BR-50, BR-51, BR-52
Other Information:	Endpoint: POST /api/auth/register
Assumptions:	Customer is the author of the like.

2.3.140 View blog detail

UC ID and Name:	**UC-59 View blog detail**		
Create By:	Tran Minh Thang	Date Created:	06/06/2025
Primary actor:	User, Customer	Secondary Actors:	None
Trigger:	The user clicks on a specific blog in the blog list.		
Description:	This use case allows users to view detailed information about a selected blog.		
Preconditions:	The user has accessed the blog list.		
Postconditions:	Detailed information about the blog is displayed.		

Normal flow:	**59.0 View blog detail** 1. The user selects a blog from the list. 2. System retrieves blog data. 3. System displays full detail of the selected blog.
Alternative Flows:	N/A
Exceptions:	**59.0.E1 System error or API failure** Inform “Unable to load data. Please try again later.”
Priority:	Medium
Frequency of Use:	Regularly
Business Rules:	N/A
Other Information:	Uses endpoint /api/blog/{id} to fetch data.
Assumptions:	Blog ID is valid and exists in the system.

2.3.141 View list of posts

UC ID and Name:	**UC-60.1 View list of posts**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin access to "Quản lý mạng xã hội" menu item in management page.		
Description:	This feature allows the Admin to view all posts in the system.		
Preconditions:	Admin is logged into the system. Posts exist in the database.		
Postconditions:	The list of posts is displayed to the Admin.		
Normal flow:	**60.1 View posts** 1. Admin selects the managed reservations menu. 2. The system retrieves all reservations from the database. 3. The system displays the list of reservations.		
Alternative Flows:	N/A		
Exceptions:	**60.1.E1 No posts found** Inform: "Không có bài đăng nào."		
Priority:	Medium		
Frequency of Use:	Often		
Business Rules:	N/A		
Other Information:	Endpoint: GET /api/posts		
Assumptions:	Posts are correctly stored and retrievable from the database.		

2.3.142 Modify status post

UC ID and Name:	**UC-60.2 Modify status reservation**		
Create By:	Tran Minh Thang	Date Created:	06/11/2025
Primary actor:	Admin	Secondary Actors:	None
Trigger:	Admin selects the option to update a post from the post list.		
Description:	This feature allows the Admin to update the status of an existing post (approve, reject).		
Preconditions:	Admin is logged into the system. Posts exist in the database.		
Postconditions:	If Approved: The post becomes visible to all users on the platform. If Rejected: The post remains invisible to other users, and the author is notified of rejection.		
Normal flow:	**60.2 Update post** 1. The system displays the list of posts with their current status. 2. Admin selects a post and chooses a new status (approved/rejected). 3. Admin clicks the "Update" button. 4. The system updates the post status in the database and shows a success message.		
Alternative Flows:	N/A		
Exceptions:	**60.2.E1 Order not found** Inform the Admin: "Bài viết không tồn tại."		
Priority:	Medium		
Frequency of Use:	Often		
Business Rules:	BR-42, BR-43, BR-48		
Other Information:	Endpoint: PUT /api/posts/{id}/status		
Assumptions:	Admin has the necessary permissions to update post statuses.		

3. Functional Requirements

3.1 System Functional Overview

3.1.1 Screens Flow

a) Guest:

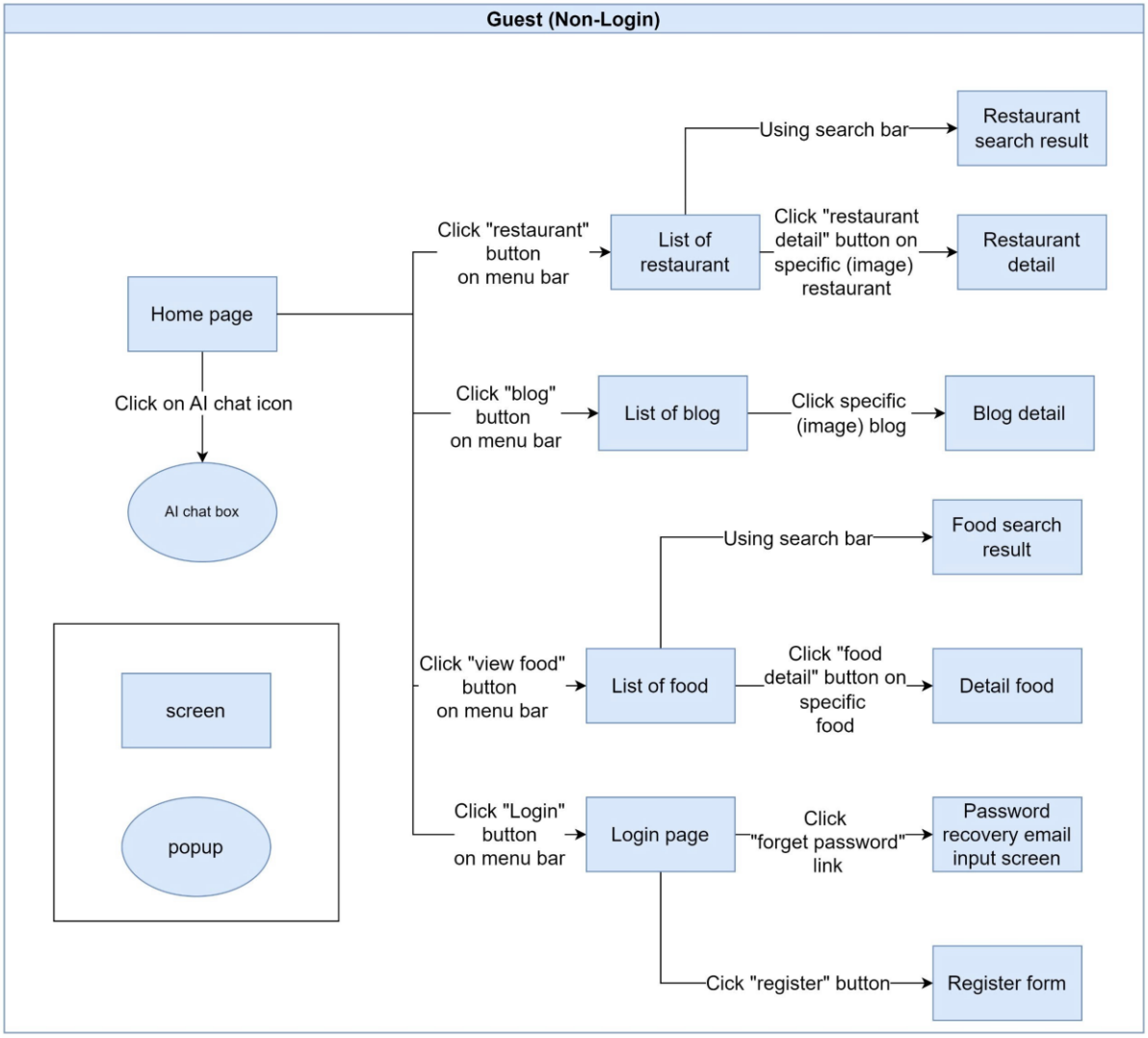


Figure 32: Screen Flow of Guest (Non-Login)

b) Customer:

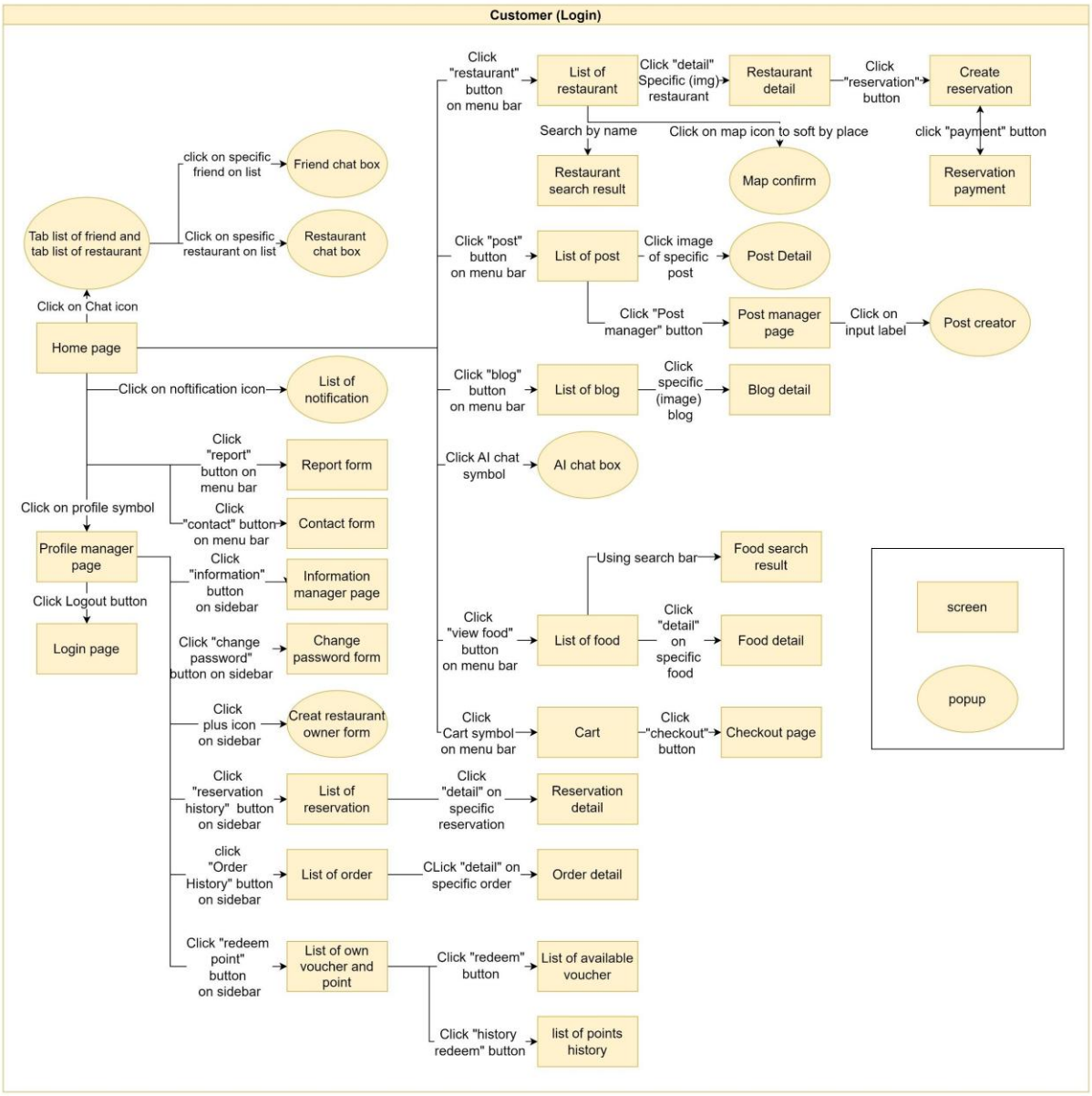


Figure 33: Screen Flow of Customer (Login)

c) Restaurant Owner:

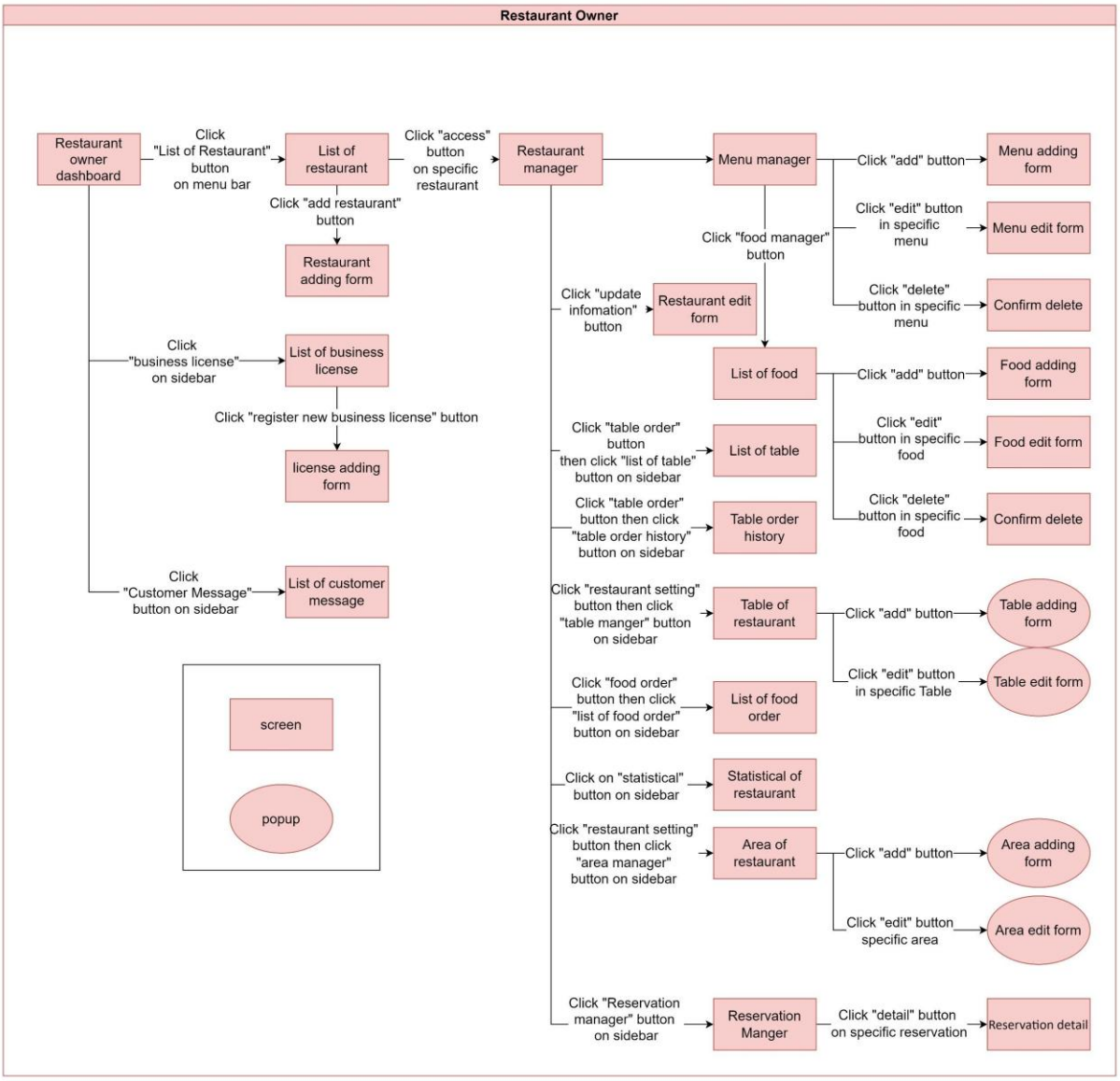


Figure 34: Screen Flow of Restaurant Owner

d) Admin:

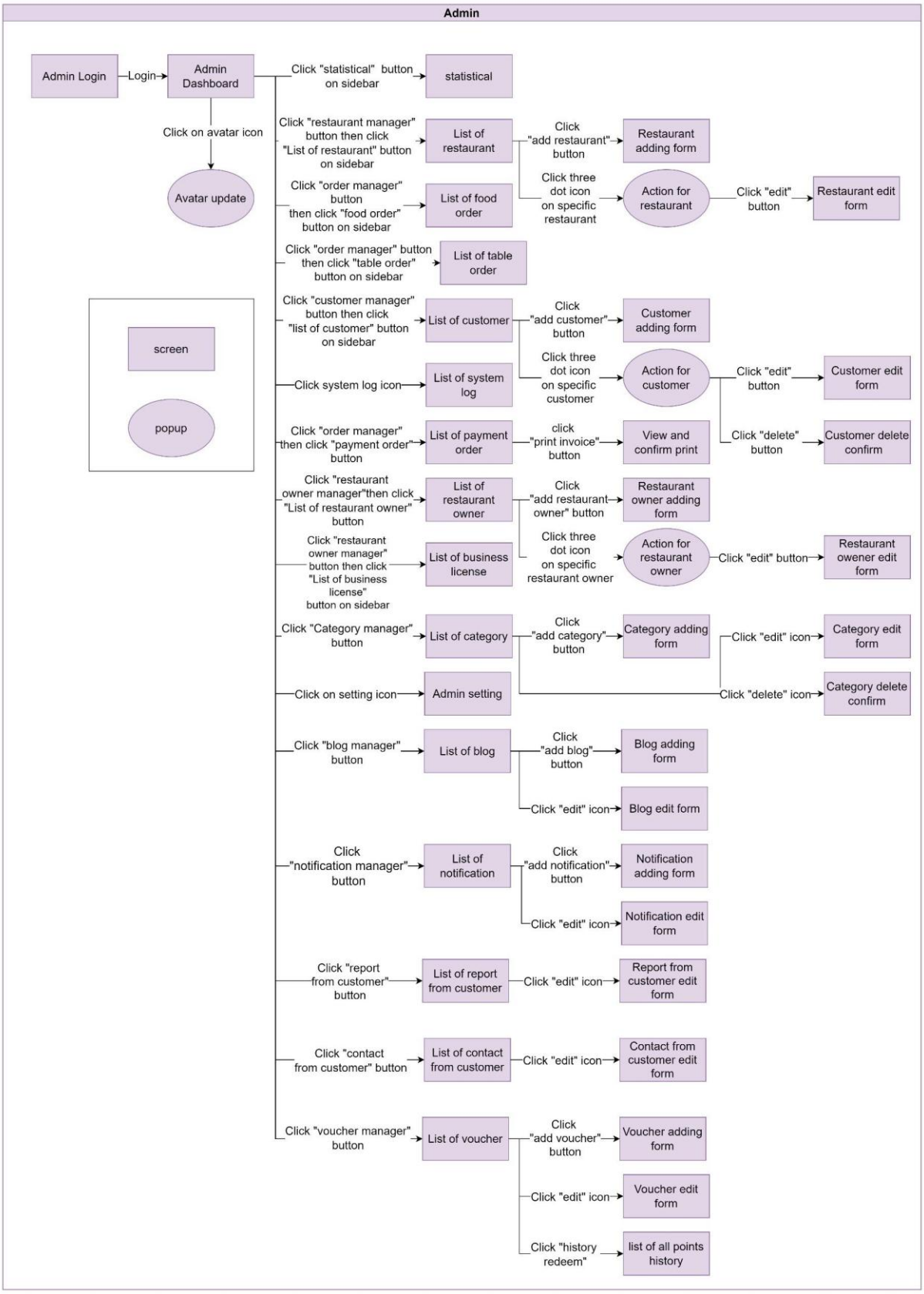


Figure 35: Screen Flow of Admin

3.1.2 Screen Descriptions

To clarify system functionality and user interaction flows, the following table provides descriptions of key user interface screens. These screens represent core features accessible to both guests and authenticated users, and serve as the foundation for UI/UX design, navigation flow, and functional implementation.

Table 15: Screen Descriptions for Guest and Authentication Features

#	Feature	Screen	Description
1	View list of restaurants	List of restaurant	Display all restaurants for guests
2	View restaurant detail	Restaurant detail	Detailed restaurant view including menu and rating
3	Create Reservation	Create reservation	Create reservation to order table
4	Create Reservation	Reservation payment	To pay for a table reservation
5	View List of Posts	List of post	Public social posts
6	View List of Posts	Post detail	Full post content and comments
7	Manager's Customer Post	Post manager page	Manage own post
8	Create Posts	Post creator	Create new post
9	View List of Blog	List of blog	Blog articles for guests
10	View List of Blog	Blog detail	Blog content
11	Find Restaurants	Restaurant search result	Filter restaurants by name, location, or price

#	Feature	Screen	Description
12	View List of Foods	List of food	List of available dishes
13	View Food Detail	Food detail	Detailed food description
14	Login	Login page	Login interface for customers
15	Forget Password	Password recovery email input	Form for password reset request
16	Login	Register form	Form to create new account
17	Filter By Region	Map Confirm	Chose to set location
18	Checkout Order	Cart	Food items added for ordering
19	Checkout Order	Checkout page	Confirm and pay for order
20	Manage Customer's Profile	Profile manager page	View and manage customer profile
21	View List of Reservations	List of reservation	Customer's reservation history
22	View List of Order Food	List of order	Customer's past orders
23	Modify Profile	Information manager page	Change personal information
24	Change Password	Change password form	Reset login password

#	Feature	Screen	Description
25	Register To Be Restaurant Owner	license adding form	Form to become restaurant owner
26	Chat With AI Assistant	AI chat box	Chatbot for food recommendation
27	Chat With Customer	Tab list of friend and tab list of restaurant	list of friend and following restaurant
28	Chat With Customer	Friend chat box	Box to chat with friend
29	Chat With Customer Restaurant Owner	Restaurant chat box	Box to chat with Restaurant owner
30	View List of Notifications	List of notification	List of notification
31	Contact DinGo	Report form	To report errors and violations
32	Contact DinGo	Contact form	Leave contact information for admin
33	Register To Be Restaurant Owner	Create restaurant owner form	Create owner
34	Search Food	Food search result	Searching food and list food
35	Viewing Food Detail	Food detail	Detail and information of food
36	Manager RO's Restaurants	Restaurant owner dashboard	Restaurant owner page

#	Feature	Screen	Description
37	Register To Be Restaurant Owner	List of business license	List all pending approve license list
38	Chat With Customer	List of customer message	list customer and message
39	Viewing Restaurant owner	Restaurant manager	Page to manage restaurant
40	Manager RO's Menus	Menu manager	List menu of restaurant
41	Create Menu	Menu adding form	Create new menu
42	Modify Menu	Menu edit form	Edit menu
43	Delete Menu	Confirm delete	Delete menu
44	View List of Area	Area of restaurant	List of area
45	Create Area	Area adding form	Create new area
46	Edit Area	Area edit form	Edit Area
47	Manager RO's Restaurants	List of restaurant	List of owned restaurants
48	Modify Restaurant	Restaurant edit form	Edit restaurant info

#	Feature	Screen	Description
49	Create Restaurant	Restaurant adding form	Add a new restaurant
50	Manage Table of Restaurant	Table of restaurant	View tables per restaurant
51	Edit Table	Table edit form	Edit table info
52	Create Table	Table adding form	Add a new table
53	Viewing List of Food	List of food	List all food in menu
54	Modify Food	Food edit form	Edit food item info
55	Delete Food	Confirm delete	Delete food
56	Create Food	food adding form	Add a new food
57	View List of Table	List of table	List all table on restaurant
58	View List of Order	Table order history	Customer table order
59	View Food Order Foods	List of food order	List all food customer order
60	Manager RO's Reservation	Reservation Manger	See detail of specific food order
61	View Reservation	Reservation detail	Reservation detail
62	Manage Voucher	List of voucher	List all vouchers

#	Feature	Screen	Description
63	Create Voucher	Voucher adding form	Create new voucher
64	Modify Voucher	Voucher edit form	Modify existing voucher
65	View List of Customers	List of customer	See list of customers
66	Create Customer	Customer adding form	Create new customer by admin
67	Modify Customer	Customer edit form	Edit customer by admin
68	Delete Customer	Customer delete confirm	Delete customer by admin
69	View Revenue by Restaurant	Statistical of restaurant	Revenue overview
70	Login	Admin login	Admin access page
71	View Dashboard	Admin dashboard	Admin main interface
72	Manage Restaurant	List of restaurant	All restaurants overview (admin)
73	Create Restaurant	Restaurant adding form	Admin create new restaurant
74	Modify Restaurant	Restaurant edit form	Edit Restaurant by admin

#	Feature	Screen	Description
75	Manage License Restaurant	List of business license	Pending verification requests
76	Manager Profile	Admin setting	Setting for admin page
77	View List of Logs	List of system log	Log of all system
78	Manage Notifications	List of notification	List of system notifications
79	Create Notification	Notification adding form	Create new system message
80	Modify Notification	Notification edit form	Edit notification
81	View Dashboard	statistical	Web business revenue
82	View List of ROs	List of restaurant owner	List of all restaurant owner on system
83	Create ROs	Restaurant owner adding form	Create restaurant owner by admin
84	Modify ROs	Restaurant owner edit form	Edit restaurant owner
85	View List of Category	List of category	List all category on system

#	Feature	Screen	Description
86	Create Category	Category adding form	Create new category
87	Modify Category	Category edit form	Edit category
88	View Blog	List of Blog	List all blog on system
89	Create Blog	Blog adding form	Create new blog
90	Modify Blog	Blog edit form	Edit blog
91	View List of Reports	List of Report	List all report on system
92	Delete Report	Reportdelete confirm	Delete blog
93	View List of Contacts	List of contact from customer	List all contact on system
94	Delete Contact	Contact from customer delete confirm	Delete contact
95	View list of own vouchers and points	List of own voucher and point	View list of own voucher and point

#	Feature	Screen	Description
96	View list of available vouchers	List of available voucher	View available voucher
97	View list of own points history	List of own points history	View own point history
98	View list of points history	List of all points history	View all customer point history

3.1.3 Screen Authorization

To enforce access control and ensure that each user interacts with the system within their defined scope, the following matrix outlines screen-level authorization across all user roles. This mapping helps define boundaries between general users, registered customers, restaurant owners, and system administrators, thus enhancing security and minimizing unauthorized actions.

Table 5: Screen Authorization Matrix by Guest Role

Screen	Guest	Customer	Restaurant Owner	Admin	Screen Type
List of restaurant	x	x	x	x	Screen
Restaurant detail		x	x		Screen
Create reservation		x	x		Screen
Reservation payment		x	x		Screen
List of post		x	x		Screen
Post detail		x	x		Pop-Up

Screen	Guest	Customer	Restaurant Owner	Admin	Screen Type
Post manager page		x	x		Screen
Post creator		x	x		Pop-Up
List of blog		x	x		Screen
Blog detail		x	x		Screen
Restaurant search result		x	x		Screen
List of food	x	x	x		Screen
Food detail	x	x	x		Screen
Login page	x	x	x		Screen
Password recovery email input	x				Screen
Register form	x				Screen
Map Confirm	x	x	x		Screen
Cart		x	x		Screen
Checkout page		x	x		Screen
Profile manager page		x	x		Screen
List of reservation		x	x		Screen
List of order		x	x		Screen
Information manager page		x	x		Screen
Change password form		x	x		Screen
license adding form			x		Screen

Screen	Guest	Customer	Restaurant Owner	Admin	Screen Type
AI chat box	x	x	x		Pop-Up
Tab list of friend and tab list of restaurant		x	x		Pop-Up
Friend chat box		x			Pop-Up
Restaurant chat box		x			Pop-Up
List of notification		x			Pop-Up
Report form		x	x		Screen
Contact form		x	x		Screen
Create restaurant owner form		x	x		Screen
Food search result	x	x	x		Screen
Food detail	x	x	x		Screen
Restaurant owner dashboard			x		Screen
List of business license			x		Screen
List of customer message			x		Screen
Restaurant manager			x		Screen
Menu manager			x		Screen
Menu adding form			x		Screen
Menu edit form			x		Screen
Confirm delete (menu)			x		Screen

Screen	Guest	Customer	Restaurant Owner	Admin	Screen Type
Area of restaurant			x		Screen
Area adding form			x		Pop-Up
Area edit form			x		Pop-Up
List of restaurant			x		Screen
Restaurant edit form			x		Screen
Restaurant adding form			x		Screen
Table of restaurant			x		Screen
Table edit form			x		Pop-Up
Table adding form			x		Pop-Up
List of food			x		Screen
Food edit form			x		Screen
Confirm delete (food)			x		Screen
food adding form			x		Screen
List of table			x		Screen
Table order history			x		Screen
List of food order			x		Screen
Reservation Manger			x		Screen
Reservation detail			x		Screen
List of voucher				x	Screen

Screen	Guest	Customer	Restaurant Owner	Admin	Screen Type
Voucher adding form				x	Screen
Voucher edit form				x	Screen
List of customer				x	Screen
Customer adding form				x	Screen
Customer edit form				x	Screen
Customer delete confirm				x	Screen
Statistical of restaurant			x		Screen
Admin login				x	Screen
Admin dashboard				x	Screen
List of restaurant (admin)				x	Screen
Restaurant adding form (admin)				x	Screen
Restaurant edit form (admin)				x	Screen
List of business license				x	Screen
Admin setting				x	Screen
List of system log				x	Screen
List of notification				x	Screen
Notification adding form				x	Screen
Notification edit form				x	Screen

Screen	Guest	Customer	Restaurant Owner	Admin	Screen Type
statistical				x	Screen
List of restaurant owner				x	Screen
Restaurant owner adding form				x	Screen
Restaurant owner edit form				x	Screen
List of category				x	Screen
Category adding form				x	Screen
Category edit form				x	Screen
List of Blog				x	Screen
Blog adding form				x	Screen
Blog edit form				x	Screen
List of Report				x	Screen
Reportdelete confirm				x	Screen
List of contact from customer				x	Screen
Contact from customer delete confirm				x	Screen
List of own voucher and point		x	x		Screen
List of available voucher		x	x		Screen
list of points history		x	x		Screen

Screen	Guest	Customer	Restaurant Owner	Admin	Screen Type
list of all points history				x	Screen

3.1.4 Non-Screen Functions

In addition to the user interface functionalities, the DineGO system includes several non-screen background services that play a critical role in ensuring seamless operations, automation, and system reliability. These functions operate independently of user interaction and are executed through scheduled jobs, system triggers, or background processes. The table below summarizes the key non-screen functions implemented in the platform.

Table 16*: Non-Screen System Functions of the DineGO Platform*

#	Feature	System Function	Description
1	Reservation	Auto-cancel Unconfirmed Booking	Automatically cancels a reservation if the user does not confirm within a predefined time limit (e.g., 15 minutes)>
2	Notification	Send Notification Job	A scheduled task that sends reminders for upcoming reservations, new promotions, or social interactions.
3	Invoice	Invoice Generator	Generates an electronic invoice and emails it to the customer after a successful VNPay payment
4	Analytics	Customer Behavior Analytics	Background service that analyzes booking history, peak times, and user preferences to assist restaurants in service optimization.
5	DB Optimization	Booking Conflict Resolver	Handles booking conflicts (e.g., double booking the same table) using transaction and rollback mechanisms.
6	AI Recommendation (Optional)	Meal Recommendation Engine	Recommends meals based on user history and dietary goals (e.g., keto, eat clean).
7	Security	Token Expiry Cleaner	Periodically clears expired JWT or session tokens from the system.
8	Map	Google Maps Routing API Handler	Service that queries Google API to get the route from the user's location to the selected restaurant.
9	Social Feed	Video Transcode Service	Processes and compresses uploaded videos before storage or streaming on the in-app social media feature

3.1.5 Entity Relationship Diagram

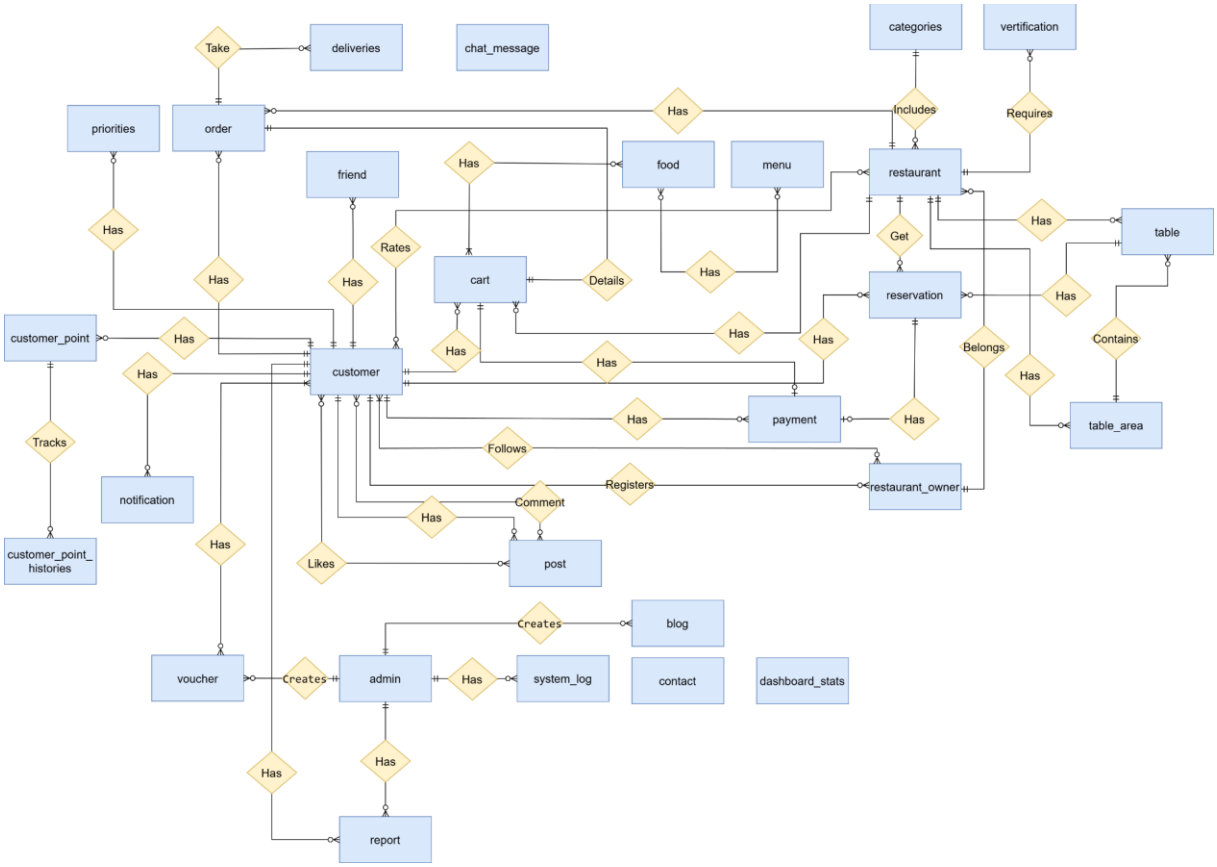


Figure 36: Entity Relationship Diagram of DineGO System

admin entity

Attribute	Data Type	PK	FK	Null
ad_id	int	X		
ad_username	nvarchar(50)			
ad_password	nvarchar(255)			
ad_name	nvarchar(255)			
ad_email	nvarchar(100)			
ad_birthday	datetime2(3)			
ad_image	nvarchar(255)			X
ad_is_use	bit			X
admin_is_deleted	bit			

customer entity

Attribute	Data Type	PK	FK	Null
cus_id	int	X		
cus_username	nvarchar(50)			
cus_password	nvarchar(255)			
cus_name	nvarchar(100)			
cus_email	nvarchar(100)			
cus_phone	nvarchar(20)			
cus_address	nvarchar(500)			
cus_birthday	datetime2(3)			X
cus_gender	bit			X
cus_image	nvarchar(300)			X
cus_is_kyi	bit			X
google_id	nvarchar(100)			X
login_provider	nvarchar(20)			X
cus_status	int			X
cus_created_date	datetime			X
cus_last_login_date	datetime			X
cus_is_use	bit			
cus_is_deleted	bit			
cus_latitude	float			
cus_longitude	float			

restaurant_owner entity

Attribute	Data Type	PK	FK	Null
res_owner_id	int	X		
cus_id	nvarchar(50)		X	
res_owner_name	nvarchar(255)			
res_owner_created_date	nvarchar(100)			
res_owner_follower_count	nvarchar(100)			

Attribute	Data Type	PK	FK	Null
res_owner_is_use	nvarchar(20)			
res_owner_is_deleted	nvarchar(500)			

restaurant entity

Attribute	Data Type	PK	FK	Null
res_id	int	X		
cate_id	int		X	
res_owner_id	int		X	
res_name	nvarchar(500)			
res_address	nvarchar(200)			X
res_email	nvarchar(255)			X
res_phone	nvarchar(20)			X
res_description	nvarchar(max)			X
res_rate	decimal(3,2)			X
res_reservation_fee	decimal(10,2)			X
res_discount_promotion	decimal(5,2)			X
res_images	nvarchar(max)			X
res_is_use	bit			
res_rate_count	int			
res_is_deleted	bit			
res_latitude	float			X
res_longitude	float			X
res_is_authorize	bit			

categories entity

Attribute	Data Type	PK	FK	Null
cate_id	int	X		
cate_type	nvarchar(50)			
cate_description	nvarchar(200)			X

Attribute	Data Type	PK	FK	Null
cate_is_deleted	bit			

menu entity

Attribute	Data Type	PK	FK	Null
menu_id	int	X		
res_id	int		X	
menu_type	nvarchar(20)			
menu_image	nvarchar(300)			
menu_name	nvarchar(20)			
menu_is_deleted	bit			

food entity

Attribute	Data Type	PK	FK	Null
food_id	int	X		
menu_id	int		X	
food_name	nvarchar(100)			
food_description	nvarchar(200)			
food_price	decimal(8,2)			
food_image	nvarchar(300)			
food_status	int			
food_is_deleted	bit			X
food_tag	nvarchar(200)			

cart entity

Attribute	Data Type	PK	FK	Null
cart_id	int	X		
res_id	int		X	
cus_id	int		X	
cart_is_deleted	bit			

order entity

Attribute	Data Type	PK	FK	Null
order_id	int	X		
cus_id	int		X	
res_id	int		X	
order_date	datetime			
order_status	int			
order_total	decimal(8,2)			
order_price_discount	decimal(8,2)			X
voucher_code_applied	varchar(200)			

order_detail entity

Attribute	Data Type	PK	FK	Null
order_detail_id	int	X		
cart_id	int		X	
order_id	int		X	
order_quantity	int			
order_price	decimal(8,2)			

reservation entity

Attribute	Data Type	PK	FK	Null
reser_id	int	X		
cus_id	int		X	
res_id	int		X	
table_id	int		X	
reser_date	datetime			
reser_created_at	datetime2(7)			
reser_status	nvarchar(50)			
reser_note	nvarchar(max)			
reser_is_deleted	bit			

payment entity

Attribute	Data Type	PK	FK	Null
pay_id	int	X		
cart_id	int		X	X
cus_id	int		X	
reser_id	int		X	X
pay_price	decimal(8,2)			
pay_status	int			X
pay_created_date	datetime			
pay_price_discount	decimal(8,2)			X
pay_is_deleted	bit			

voucher entity

Attribute	Data Type	PK	FK	Null
voucher_id	int	X		
ad_id	int		X	X
voucher_code	nvarchar			
voucher_discount	decimal(8,2)			
voucher_start_date	datetime			
voucher_end_date	datetime			
voucher_stock	int			X
voucher_is_deleted	bit			
voucher_is active	bit			X
voucher_type	int			
voucher_apply_type	int			

post entity

Attribute	Data Type	PK	FK	Null
post_id	int	X		
res_id	int		X	X
cus_id	int		X	X

Attribute	Data Type	PK	FK	Null
post_content	text			
post_image	nvarchar(max)			
post_created_date	datetime			
post_updated_date	datetime			
post_title	nvarchar(255)			X
post_author_name	nvarchar(50)			X
post_like_count	int			X
post_comment_count	int			X
post_is_approve	bit			X

blog entity

Attribute	Data Type	PK	FK	Null
blog_id	int	X		
res_owner_id	int		X	
ad_id	int		X	X
blog_title	nvarchar(150)			
blog_information	nvarchar(max)			
blog_date	datetime2(7)			
blog_image	nvarchar(max)			X
blog_is_deleted	bit			

notification entity

Attribute	Data Type	PK	FK	Null
noti_id	int	X		
noti_title	nvarchar			
noti_content	nvarchar(max)			
noti_type	nvarchar			
noti_date	datetime			
noti_schedule	datetime			

Attribute	Data Type	PK	FK	Null
noti_action	nvarchar(255)			

deliveries entity

Attribute	Data Type	PK	FK	Null
de_id	int	X		
order_id	int		X	
de_status	int			
de_start	datetime			
de_end	datetime			
de_note	nvarchar(max)			X
de_is_deleted	bit			

verification entity

Attribute	Data Type	PK	FK	Null
ver_id	int	X		
res_id	int		X	
ver_license	nvarchar(100)			
ver_license	nvarchar(50)			
ver_license	nvarchar(300)			
ver_status	int			
ver_date_submitted	datetime			
ver_date_verified	datetime			
ver_file_attachment	nvarchar(500)			X
ver_is_deleted	bit			
ver_date_responded	datetime2(7)			X
ver_content_responded	nvarchar(500)			X

system_log entity

Attribute	Data Type	PK	FK	Null
sys_log_id	int	X		
ad_id	int		X	X
action	nvarchar(100)			X
description	nvarchar(max)			X
log_time	datetime2(3)			X
ip_address	nvarchar(45)			X
device_info	nvarchar(255)			X
status_code	int			X
is_success	bit			X

report entity

Attribute	Data Type	PK	FK	Null
report_id	int	X		
cus_id	int		X	X
reported_user_id	int			X
report_type	nvarchar(50)			X
report_content	nvarchar(max)			X
report_status	int			
report_created_at	datetime2(7)			
admin_id	int			X
admin_note	nvarchar(max)			X
report_related_url	nvarchar(255)			X
report_is_deleted	bit			

contact entity

Attribute	Data Type	PK	FK	Null
contact_id	int	X		
contact_name	nvarchar(100)			X
contact_email	navarchar(150)			X

Attribute	Data Type	PK	FK	Null
contact_subject	nvarchar(200)			X
contact_message	nvarchar(max)			X
contact_status	int			
contact_reply_by_admin	nvarchar(100)			X
contact_reply_message	nvarchar(max)			X
contact_created_at	datetime			
contact_replied_at	datetime			X
contact_is_deleted	bit			

dashboard_stats **entity**

Attribute	Data Type	PK	FK	Null
id	int	X		
report_date	datetime2(7)			
revenue_by_month_json	nvarchar(max)			X
orders_by_month_json	nvarchar(max)			X
customer_type_json	nvarchar(max)			X
order_status_json	nvarchar(max)			X
top_restaurant_json	nvarchar(max)			X
service_revenue_json	nvarchar(max)			X
customer_group_json	nvarchar(max)			X
created_at	datetime2(7)			

table entity

Attribute	Data Type	PK	FK	Null
table_id	int	X		
res_id	int		X	
table_name	nvarchar(100)			
area_id	int		X	
table_seat	int			

Attribute	Data Type	PK	FK	Null
table_image	nvarchar(255)			X
table_position_x	int			
table_position_y	int			
table_is_deleted	bit			
table_created_at	datetime2(7)			
table_update_at	datetime2(7)			X
table_status	int			

table_area entity

Attribute	Data Type	PK	FK	Null
area_id	int	X		
res_id	int		X	
area_name	nvarchar(100)			
area_description	nvarchar(255)			X
is_deleted	bit			
created_at	datetime2(7)			
updated_at	datetime2(7)			X

friend entity

Attribute	Data Type	PK	FK	Null
friend_id	int	X		
customer_id	int		X	
friend_customer_id	int			
created_at	datetime2(7)			

chat_message entity

Attribute	Data Type	PK	FK	Null
chat_id	int	X		
sender_id	int			
receiver_id	int			

Attribute	Data Type	PK	FK	Null
message	varchar			
sent_at	datetime2(7)			
is_read	bit			

priorities entity

Attribute	Data Type	PK	FK	Null
priority_id	int	X		
cus_id	int		X	
tag	nvarchar(50)			
count	int			
click_count	int			
last_used	datetime2(7)			X
score	float			
weight_manual	float			X

customer_point entity

Attribute	Data Type	PK	FK	Null
point_id	int	X		
cus_id	int		X	
point_balance	int			
created_date	datetime			
last_updated	datetime			

customer_point_histories entity

Attribute	Data Type	PK	FK	Null
history_id	int	X		
point_id	int		X	
change_amount	int			

Attribute	Data Type	PK	FK	Null
balance_after	int			
description	nvarchar(max)			
created_date	datetime2(7)			

This table provides an overview of the main entities used in the DineGO system. Each entity represents a core business object within the platform and includes a short description of its role and responsibilities in the system.

Table 7: Entity Descriptions of the DineGO System

#	Entity	Description
1	customer	Represents the end-user who can register, log in, reserve tables, place food orders, interact with posts, and follow restaurants or other users.
2	restaurant_owner	A special type of user who registers to manage a restaurant profile, respond to reservations, manage menus, and interact with customers.
3	restaurant	Contains detail about a dining location including name, address, category, verification status, and owner.
4	reservation	Records a customer's table booking at a restaurant for a specific date and time.
5	order	Represents a food order placed by a customer. May include delivery and payment information.
6	food	A single menu item offered by a restaurant. Can be part of multiple menus.
7	menu	A collection of food items grouped for a restaurant.
8	cart	Temporary food selection (cart) before placing an order.
9	deliveries	Tracks the shipping/delivery process of an order to the customer.
10	notification	General system message template (promotion, reminder, etc.).

#	Entity	Description
11	voucher	Promotional code or discount offered by the admin.
12	post	Social media-style post created by users (photos, reviews, etc.).
13	blog	Article or promotional content written by restaurant owners.
14	categories	Type of cuisine or restaurant (e.g., Asian, Vegan).
15	verification	Document or process status indicating a restaurant's legitimacy.
16	admin	System administrator with elevated privileges to manage users, vouchers, and logs.
17	payment	Stores payment detail for customer orders/reservations, including amount, status, and date
18	friend	Represents the connection between two customers, allowing them to chat, view profiles, and interact socially
19	chat_mesage	Real-time message between users or user–restaurant; text with read status.
20	order_detail	Lists individual food items in an order, with quantity, unit price, and any applied discounts.
21	table	Physical table with code, capacity, status
22	table_area	Zone of tables with settings
23	dashboard_stats	Pre-aggregated metric snapshots for dashboards
24	contact	Contact ticket with subject, priority, status.
25	report	Analytics report with filters and export options
26	priorities	Defines priority levels
27	customer_point	Stores the current total loyalty points of each customer.
28	custome_point_histories	Stores the transaction history of point changes for each customer
29	system_log	Tracks system events, errors, and admin activities for auditing and debugging.

3.2 Authorized Account

3.2.1 Login

Function trigger: The login function is the first interaction point when accessing the DineGO system via the original URL:http://dinego.shop/Auth/Login This is the login page for all authenticated roles.

Function description: The login screen allows authorized roles to access the system, including customers, restaurant owners. The login component serves as the gateway to personalized content, and access control is enforced based on each role.

Function Detail:

➔ *Normal execution case*: When the user inputs a valid username/email and password that exist in the system, authentication is processed, and the user is redirected to their respective dashboard: customers to the booking area, restaurant owners to the management panel, and admins to the admin dashboard.

➔ *Abnormal execution case*: Login failures are classified into three types:

1. **Empty username**: The user attempts to log in without entering either or both required fields.
2. **Empty password**: The system detects that the entered username is not a valid email address.
3. **Wrong credentials**: The provided email or password does not match any existing account.

Screen layout:

The layout includes two text fields for account and password, a red "Login" button, an option to log in via Google, a “Forgot Password?” link, and a redirect for new users to register.

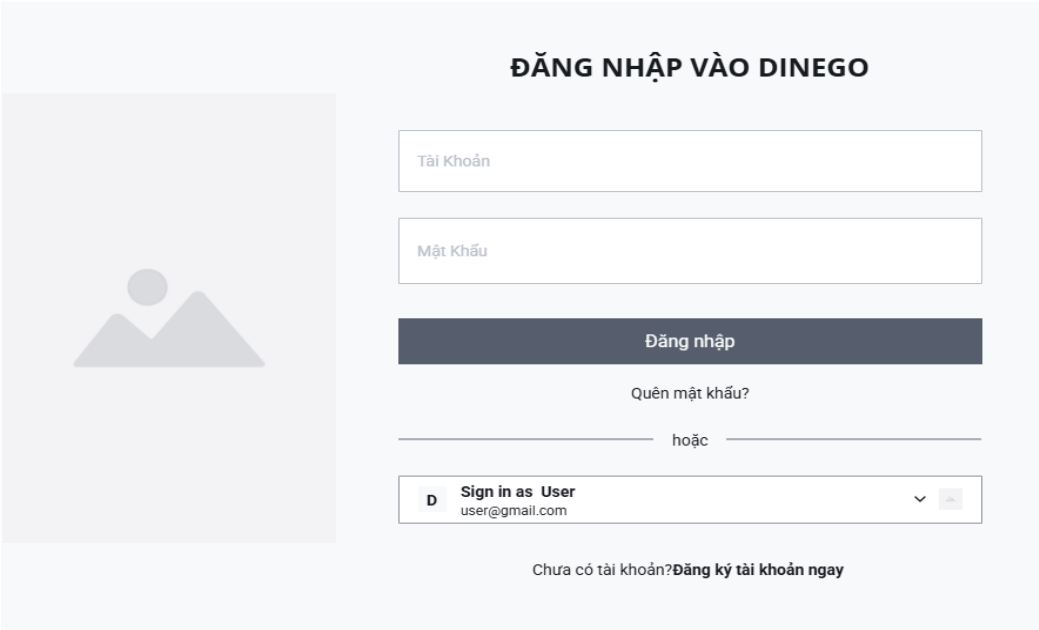


Figure 37: Customer login page of DineGO System

3.2.2 Register

Function trigger: This function is triggered when the user visits the registration link: http://dinego.shop/Auth/Register It is accessible via the "Đăng ký" (Register) link on the login page or homepage navigation bar for guests who have not yet created an account.

Function description: The registration screen allows new users to create a DineGO account by submitting personal detail including full name, username, password, email, and phone number. Successful registration provides access to personalized features such as booking, ordering, and account management.

Function Detail:

➔ *Normal execution case*: A new user completes all required fields correctly and accepts the terms

and conditions. Upon successful validation, the account is created, and the user is redirected to the login screen or logged in automatically, depending on implementation.

➔ *Abnormal execution cases* include:

- **Empty required fields**: The user submits the form without filling all mandatory fields (e.g., email, password).
- **Password mismatch**: The confirmation password does not match the original password.
- **Invalid email format**: The system detects an improperly formatted email address.
- **Guestname or email already exists**: The entered username or email is already registered in the system.
- **Unchecked terms**: The user attempts to register without agreeing to the terms and conditions.

Screen layout:

The layout contains labeled input fields for full name, username, password, password confirmation, email, and phone number, along with a checkbox to agree to terms and a "Register" button.

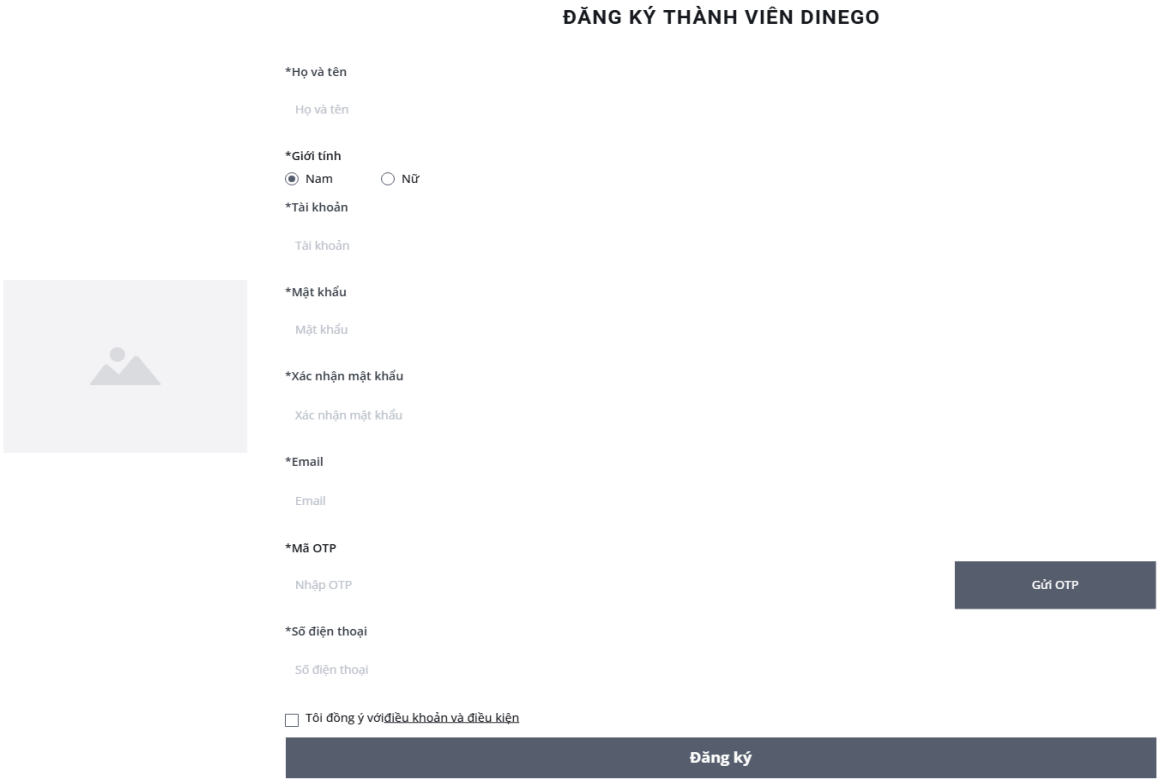


Figure 38: Guest register account page of DineGO System

3.2.3 Admin Login

Function trigger: The admin login function is triggered when an administrator accesses the dedicated admin login URL:http://admin.dinego.shop/Auth/Login

Function description: This login page allows system administrators to securely authenticate into the DineGO backend system. Unlike customers and restaurant owners, admin accounts are pre-created by the system manager and cannot self-register through the public interface. Successful login grants access to the admin dashboard and all privileged management functions.

Function Detail:

- **Normal execution case:**
  The admin enters a valid username and password → clicks the "Login" button → system verifies credentials → redirects to the Admin Dashboard.
- **Abnormal execution cases:**
  - **Empty username:** Admin submits the form without entering the username.
  - **Empty password:** Admin submits the form without entering the password.
    Wrong credentials: The system detects that the provided username or password does not match any valid admin account → shows an error message.

Screen layout:

The admin login page has:

- Guestname and Password input fields
- A secure "Login" button
- No option for public register or social login (Google OAuth is disabled for admins).

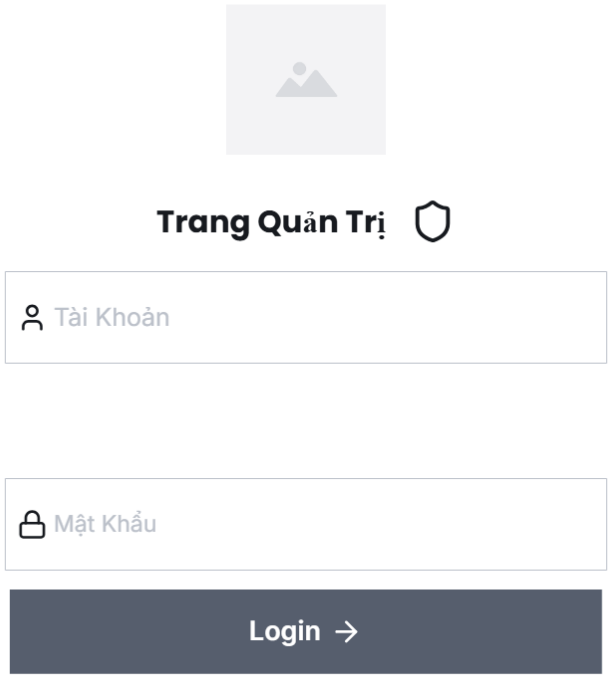


Figure 39: Admin login account page of DineGO System

3.2.4 Forgot Password

Function trigger: Guest clicks 'Forgot Password', inputs email, system sends reset link. Via the original URL: http://dinego.shop/Auth/ForgotPassword

Function description: Guest clicks 'Forgot Password', inputs email, system sends reset link.

Function Detail:

➔ Normal execution case: Guest follows valid steps as designed, system processes input and returns

result or confirmation.
➔ Abnormal execution case: If input is missing/invalid, system shows validation errors or blocks prohibited actions.
Screen layout: Includes necessary input fields, list/table views, action buttons, and feedback messages for this function.

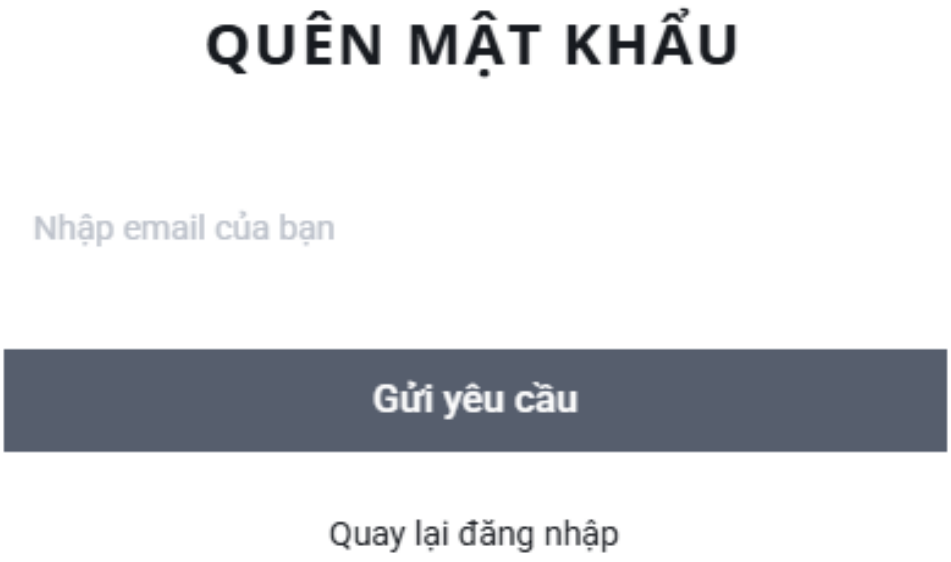


Figure 40: Forgot Password page of DineGO System

3.3 Restaurant Browsing & Detail

3.3.1 Browse Restaurants

Function trigger:

Opened when the customer accesses the "Nearby Restaurants" page from the homepage or main menu.

URL: https://dinego.shop/Restaurant/Search

Function description:

Shows restaurants by selected location with name, city/province, rating, price, and image. Guests can search, filter, and book directly.

Normal execution case:

Guest selects location → searches (optional) → views results → books a restaurant.

Abnormal execution cases:

- No location selected.
- No matching restaurants.
- Location service blocked.

Screen layout:

Location selector, search bar, search button, and restaurant cards with image, detail, and "Book Now" button.

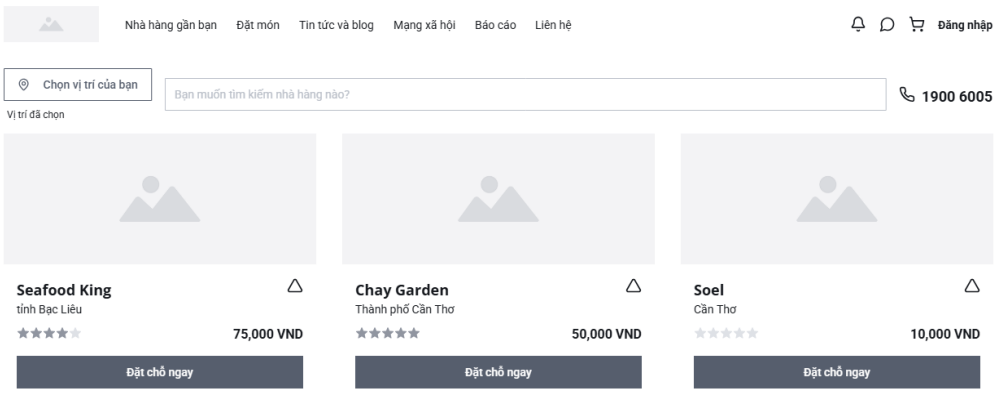


Figure 41: Browse Restaurants page of DineGO System

3.3.2 Restaurant Detail

Function trigger:

Opened when the customer clicks on a restaurant card from the Nearby Restaurants page or search results.

URL: https://dinego.shop/Restaurant/Detail/1

Function description:

Shows detail of a selected restaurant, including name, location, contact info, menu, reviews, and an option to make a reservation or follow the restaurant.

Normal execution case:

Guest opens the restaurant detail page → views info, menu, and reviews → books a table or follows the restaurant.

Abnormal execution cases:

- Restaurant not found.
- No menu available.

- Review submission restricted.

Screen layout:

Cover image, navigation tabs, restaurant info card, summary section, menu list, reviews section, and “Book Now” button.

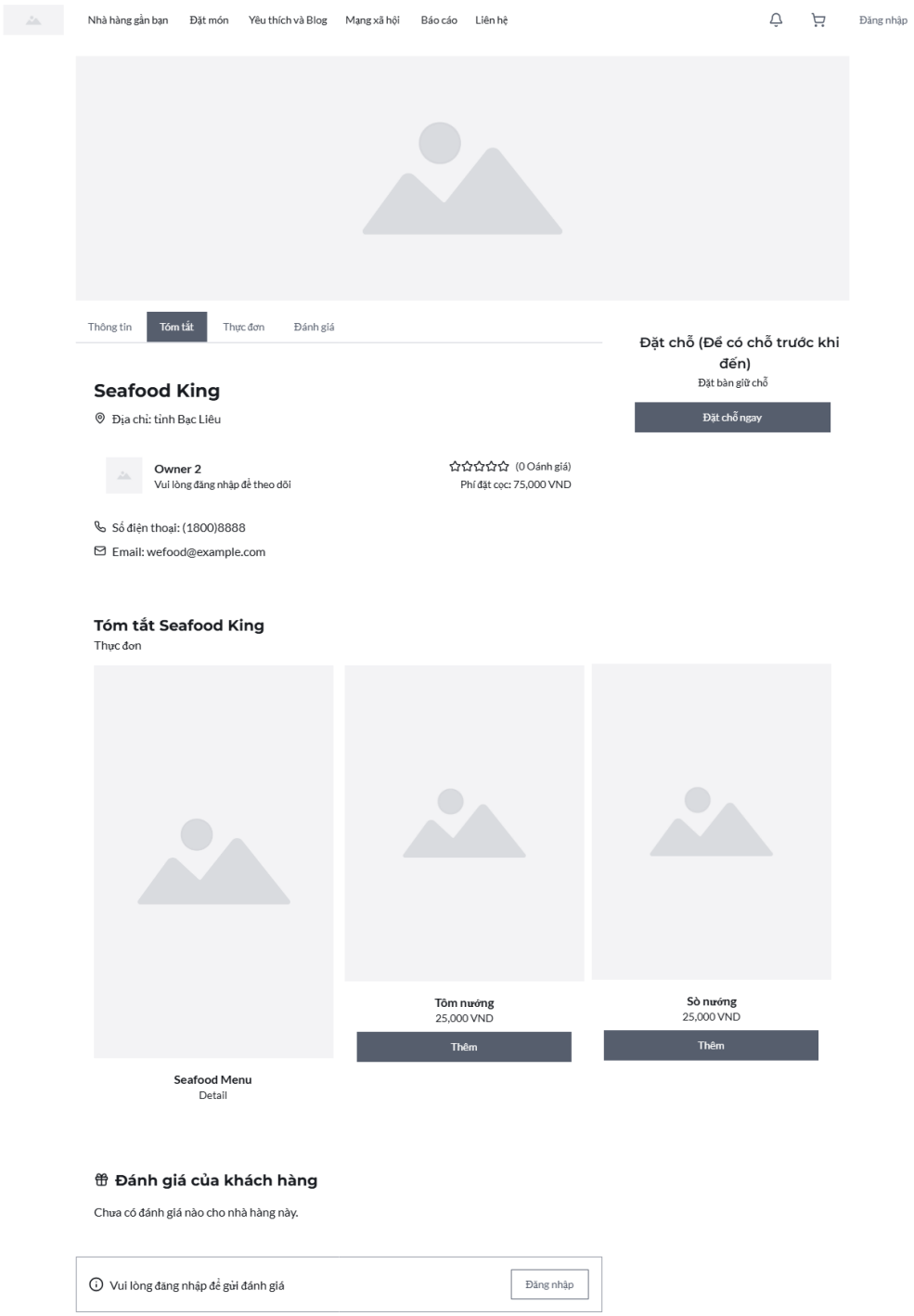


Figure 42: Restaurant Detail page of DineGO System

3.3.3 Food Viewing

Function trigger: Opened when the customer navigates to the "Food Listing" page from the main menu or homepage.

URL: https://dinego.shop/Food

Function description:

Shows a list of available food items from various restaurants with image, name, price, and quick action buttons to view detail or add to cart.

Normal execution case:

Guest opens the page → sees list of food items → views detail or adds items to cart.

Abnormal execution cases:

- No location selected.
- No available dishes.
- Add to cart failure.

Screen layout:

Location selector, search bar, and list of food cards with image, name, price, and action buttons.

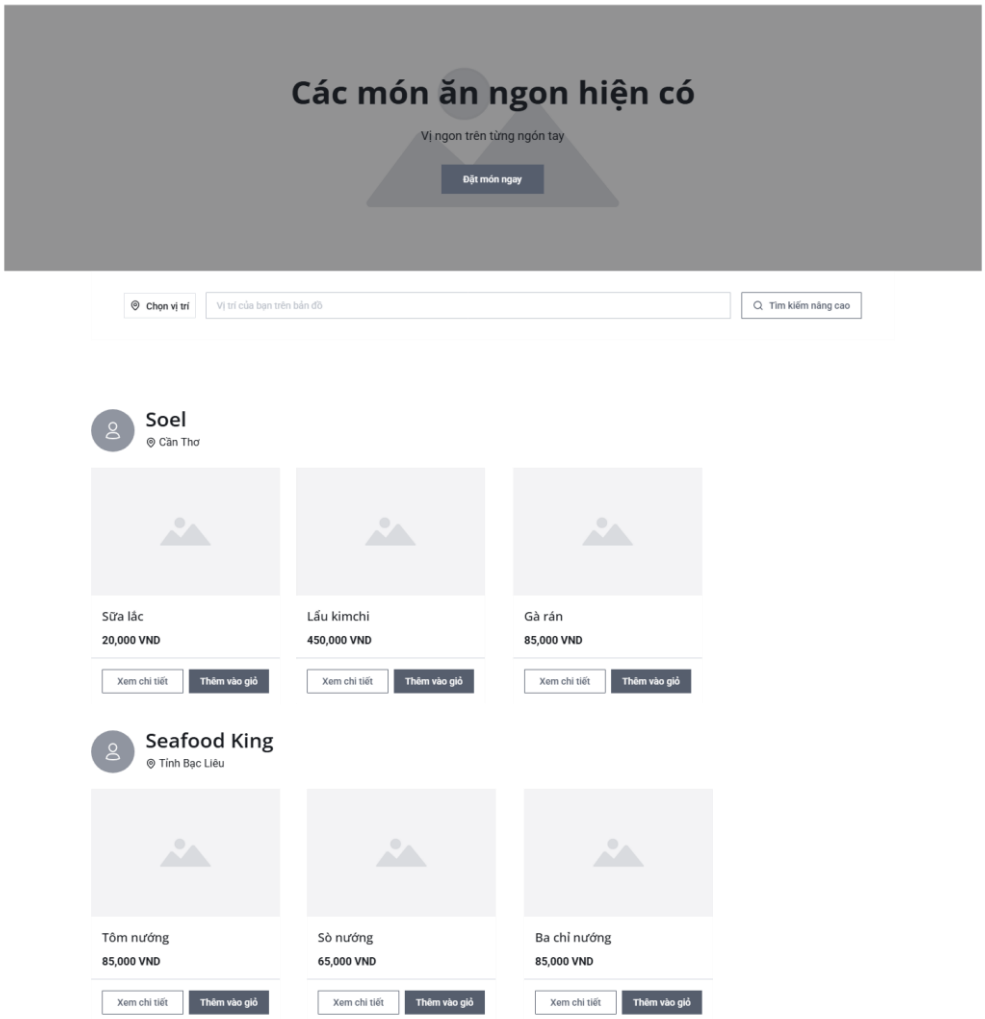


Figure 43: Food Viewing page of DineGO System

3.3.4. Food Detail

Function trigger:

Opened when the customer clicks "View Detail" on a food item from the Food Listing page. URL:https://dinego.shop/Food/Detail/3

Function description:

Shows basic information about a selected food item, including image, name, description, price, and availability. Also displays other dishes from the same restaurant and an option to add the item to the cart.

Normal execution case:

Guest opens the food detail page → views item info → adds item to cart.

Abnormal execution cases:

- Item not found.
- Item unavailable.

Screen layout:

Food image, name, description, price, "Add to Cart" button, and related dishes section.

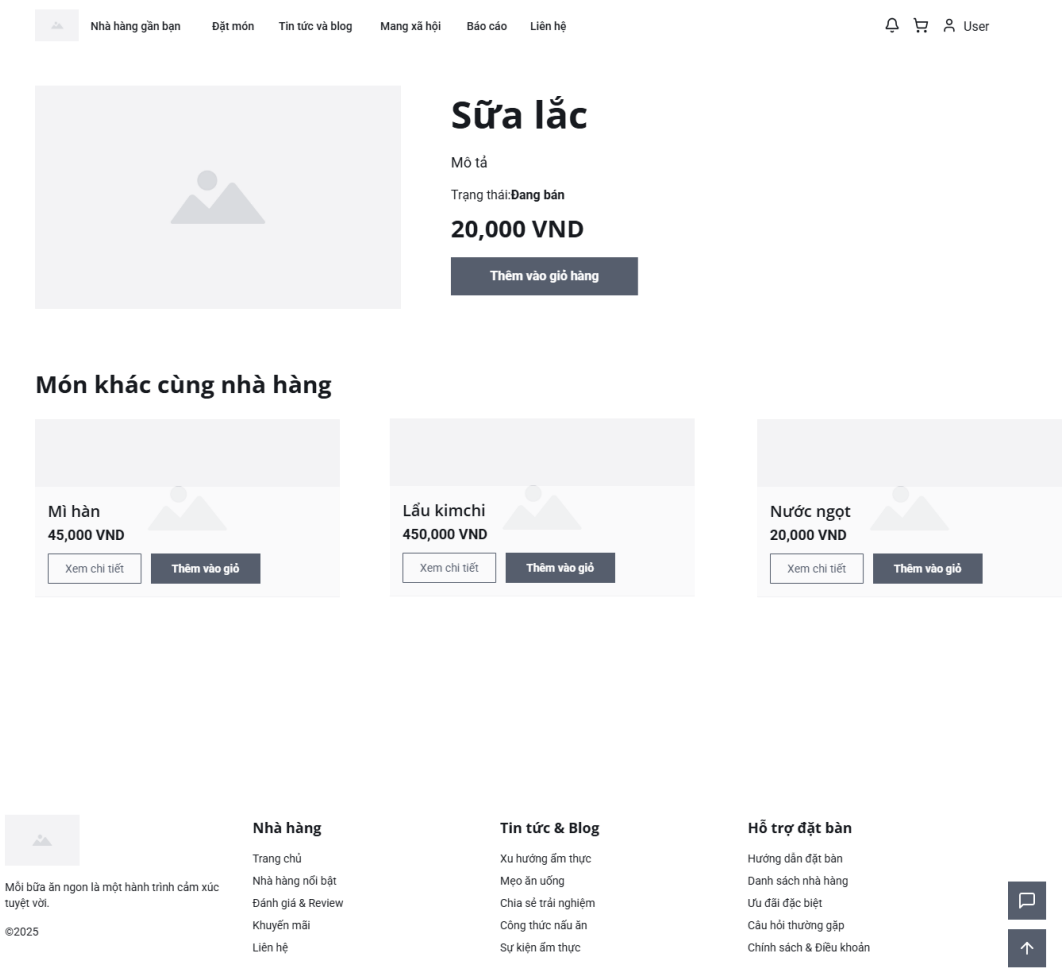


Figure 44: Food detail page of DineGO System

3.3.5 Map Location

Function trigger:

Triggered when the user clicks the **Select Location** button in a form requiring a map location (e.g., restaurant address, delivery point)..

Function description:

Displays a popup with an interactive OpenStreetMap map allowing the user to choose their location. Guests can navigate, zoom, and pinpoint the exact spot.

Normal execution case:

Guest clicks **Select Location** → popup appears with map → user selects a point → clicks **Confirm** → system saves the coordinates → popup closes.

Abnormal execution cases:

- Map fails to load → show error message.
- Current location detection fails (when using "Current Location" button) → show error message.

Screen layout:

- **Map:** OpenStreetMap with zoom in/out buttons.
- **Buttons:** Current Location, Cancel, Confirm.

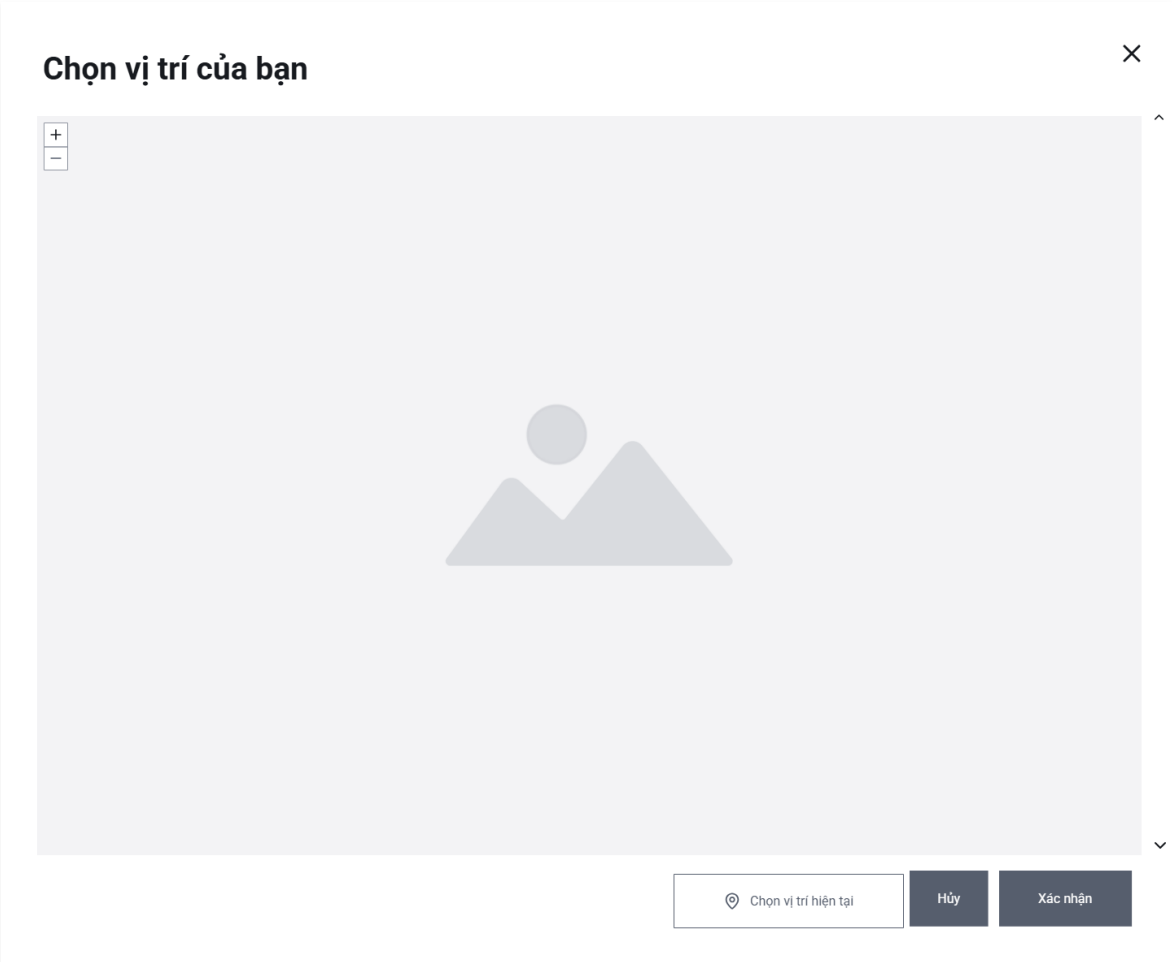


Figure 45: Map confirm of DineGO System

3.4 Ordering & Cart

3.4.1 Cart

Function trigger:

Opened when the customer clicks the cart icon or accesses Cart.

Function description:

Shows all food items the customer has added to their cart, grouped by restaurant. Allows quantity changes, item removal, and proceeding to checkout.

Normal execution case:

Guest reviews items in the cart → adjusts quantities or removes items → clicks "Checkout" to continue.

Abnormal execution cases:

- Cart is empty.
- Item unavailable or removed by the restaurant.

Screen layout:

List of items grouped by restaurant, each with name, price, quantity control, and remove option; total amount and "Checkout" button at the bottom.

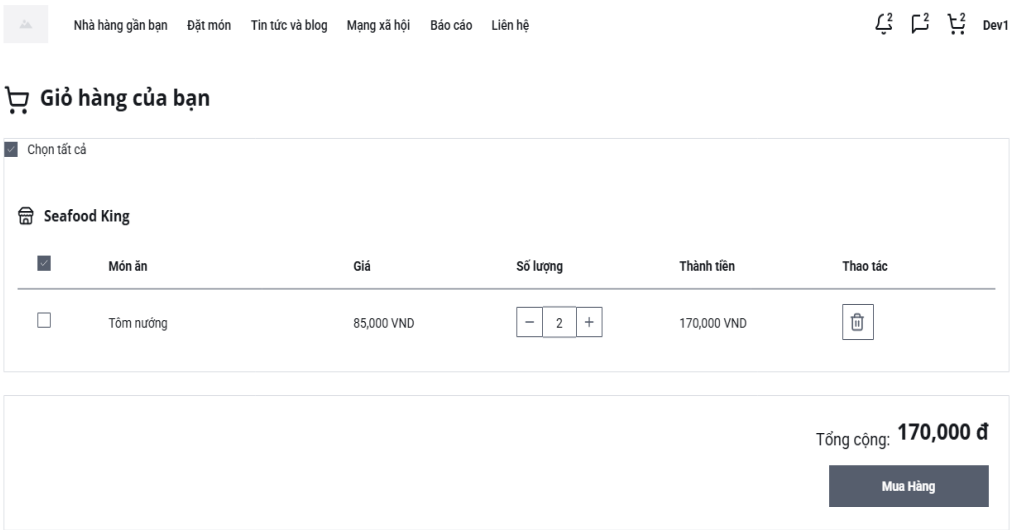


Figure 46: Cart page of DineGO System

3.4.2 Checkout

Function trigger:

Opened when the customer proceeds to checkout from the shopping cart. URL: https://dinego.shop/Cart/CheckOut

Function description:

Shows delivery detail form and order summary, including items, quantities, prices, and total amount. Allows applying a voucher and confirming the order.

Normal execution case:

Guest enters delivery information → reviews order summary → applies voucher (optional) → confirms the order.

Abnormal execution cases:

- Missing required delivery detail.
- Invalid voucher code.
- Item unavailable before confirmation.

Screen layout:

Delivery detail form, order summary table grouped by restaurant, voucher input, total price, and buttons for returning to the cart or confirming the order.

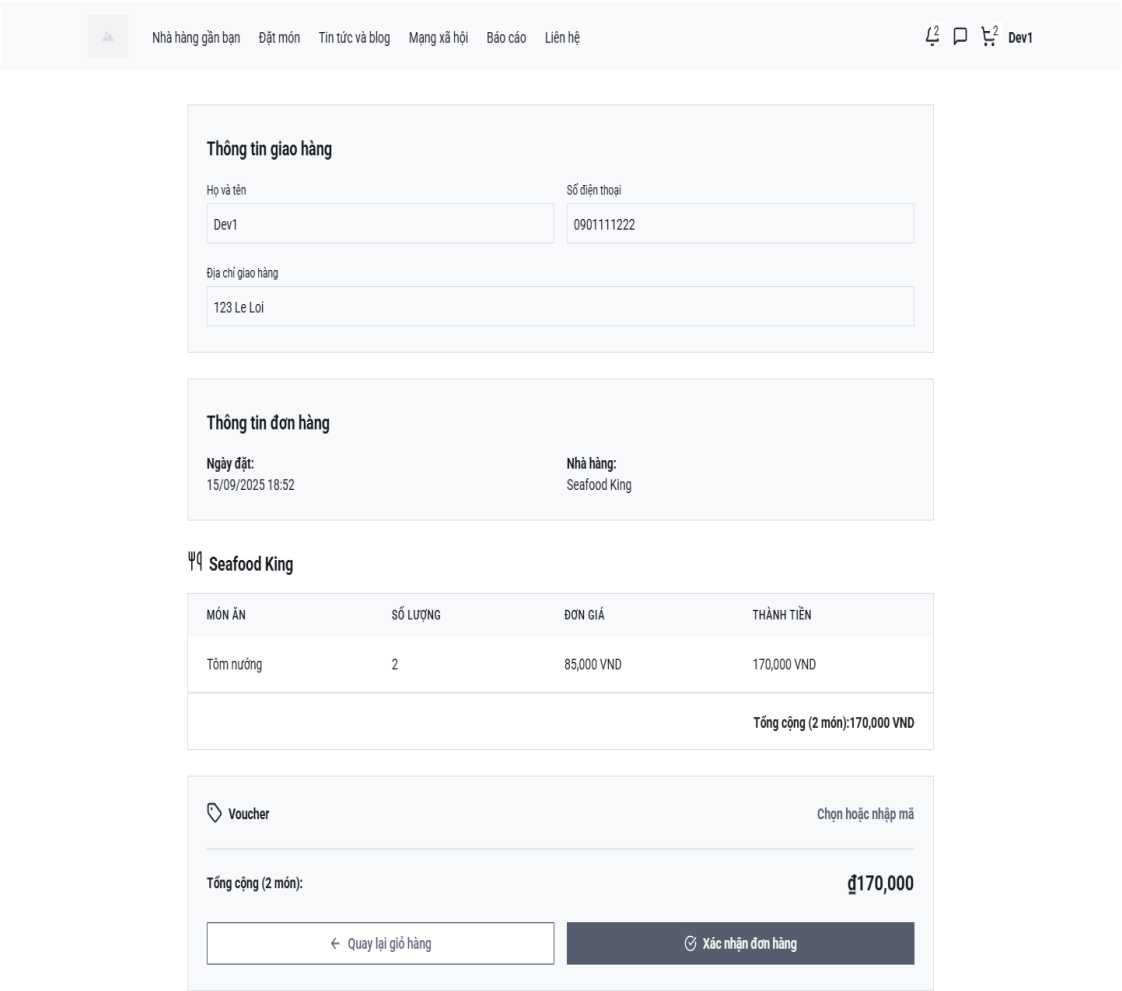


Figure 47: Checkout page of DineGO System

3.4.3 View Order History

Function trigger:

Opened when the customer selects "Track Delivery" from their account menu. URL: https://dinego.shop/Customer/TrackingDelivery

Function description:

Shows the customer's food delivery orders with status tabs (All, Pending Confirmation, Processing, Delivering, Delivered, Cancelled). Each order includes restaurant info, total amount, date, and a link to view detail.

Normal execution case:

Guest opens the tracking page → selects a status tab → views orders and their detail.

Abnormal execution cases:

- No orders found for the selected status.

Screen layout:

Status tabs, order list with restaurant name, location, date, status label, total price, and "View Order Detail" button.

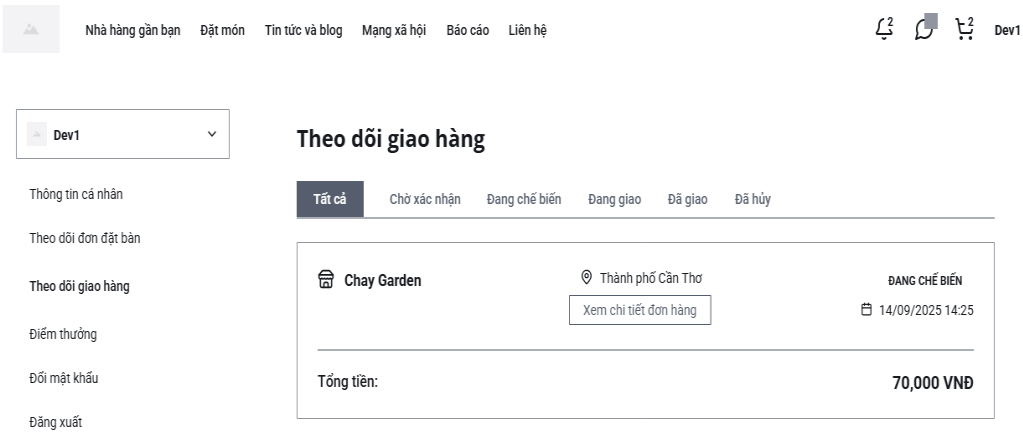


Figure 48: View Order History page of DineGO System

3.4.4 View Order Detail

Function trigger:

Opened when the customer clicks "View Order Detail" from the delivery tracking page. URL: https://dinego.shop/Customer/OrderDetail?orderId=3

Function description:

Shows a summary of a specific order, including restaurant name, pickup and delivery locations, order items, prices, voucher applied, and total cost. Provides an option to contact the restaurant.

Normal execution case:

Guest opens the order detail page → reviews order summary → contacts the restaurant if needed.

Abnormal execution cases:

- Order not found.
- Missing order information.

Screen layout:

Order header with date/time, restaurant info, pickup/delivery detail, item list with price, voucher field, total amount, and "Contact Restaurant" button.

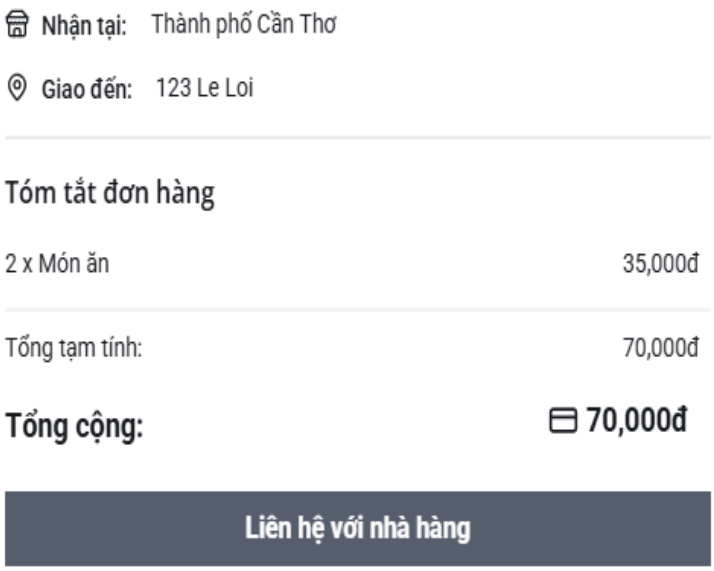


Figure 49: View Order Detail page of DineGO System

3.5 Reservation Management

3.5.1 Create Reservation

Function trigger:

Opened when the customer selects "Book Table" for a restaurant. URL: https://dinego.shop/Reservation/4

Function description:

Allows the customer to select arrival date/time, area, and table based on availability. Provides status indicators for each table and an option to leave notes before confirming payment.

Normal execution case:

Guest selects arrival time, area, and an available table → adds notes (optional) → proceeds with QR payment.

Abnormal execution cases:

- No tables available.
- Selected table becomes unavailable before confirmation.

Screen layout:

Date/time picker, area selector, table status filter, table layout with capacity, notes field, and "Pay by QR" button.

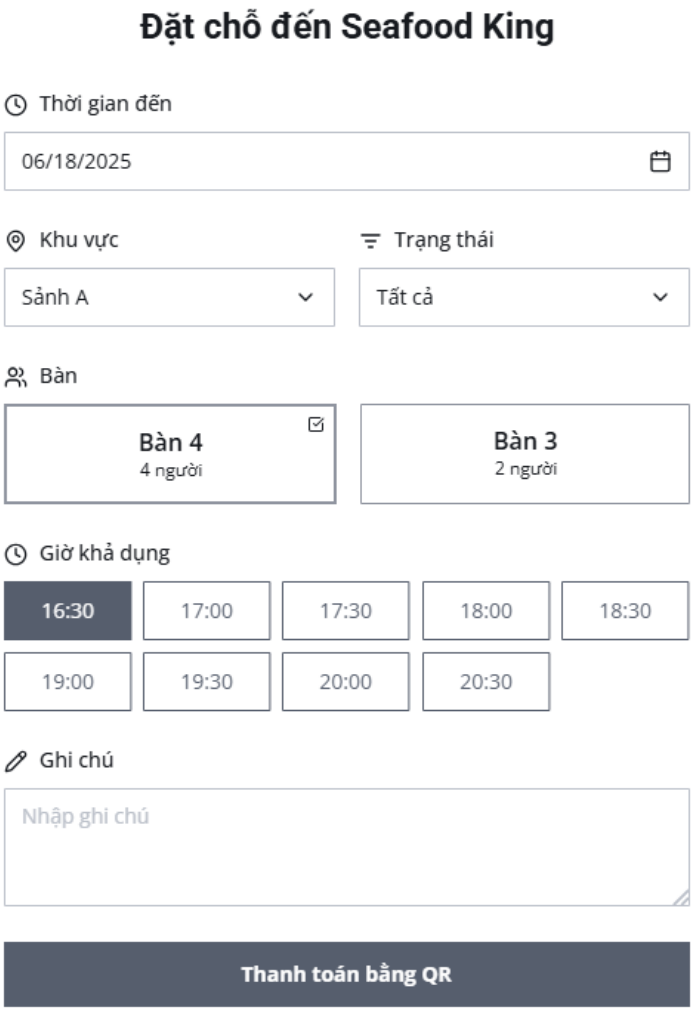


Figure 50: Create Reservation page of DineGO System

3.5.2 View Reservation

Function trigger:

Opened when the customer selects "Track Reservations" from their account menu. URL: https://dinego.shop/Customer/OrderHistory

Function description:

Displays a list of the customer's past and upcoming table reservations, including restaurant name, date/time, table, number of guests, notes, and status.

Normal execution case:

Guest opens the reservation history → reviews booking detail and status.

Abnormal execution cases:

- No reservations found.

Screen layout:

Table view with columns for restaurant, reservation time, table name, number of guests, notes, and status labels.

Lịch sử đặt bàn

Nhà hàng	Ngày đặt	Giờ đặt	Tên bàn	Số lượng	Ghi chú	Trạng thái
Seafood King	18/09/2025	18:00	Bàn 4	6 người	Yêu cầu bàn cạnh cửa sổ	Không chờ xác nhận
Seafood King	18/09/2025	13:00	Bàn 2	4 người	Không có ghi chú	Kết thúc
Nhà hàng Vịt Quay A	19/09/2025	20:00	Bàn VIP 1	8 người	Không có ghi chú	Kết thúc
Quán Lẩu Hạnh	20/09/2025	19:30	Bàn 10	5 người	Sinh nhật bạn	Không chờ xác nhận

Figure 51: View Reservation page of DineGO System

3.5.3 Payment Reservation

Function trigger:

Opened after the customer selects a table and chooses QR payment. URL: https://dinego.shop/Reservation/Payment?reser_id=3

Function description:

Displays a QR code and payment detail for the table reservation. Includes reservation information, deposit amount, and instructions for completing the transfer via mobile banking.

Normal execution case:

Guest scans the QR code using their banking app → completes the payment → reservation is confirmed.

Abnormal execution cases:

- Payment not completed before the time expires.
- Incorrect payment information.

Screen layout:

Reservation detail (restaurant, table, guests, deposit, time), QR code for payment, and step-by-step payment instructions.

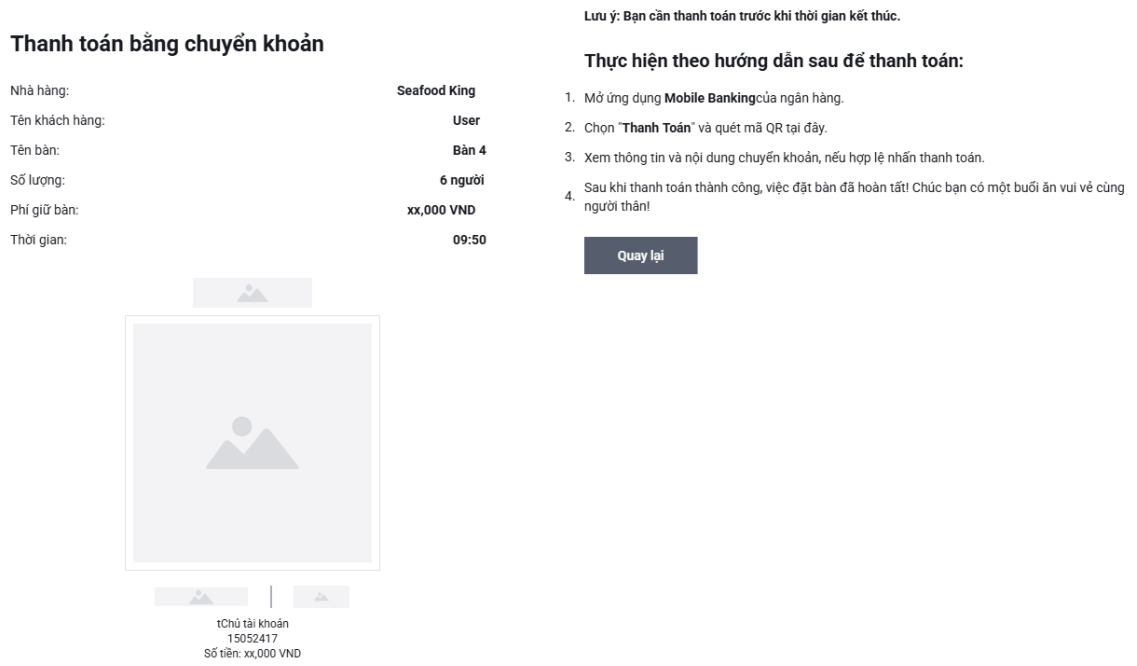


Figure 52: View Reservation payment of DineGO System

3.6 Customer Profile

3.6.1 View/Edit Profile

Function trigger:

Opened when the customer selects "Personal Information" from their account menu. URL: https://dinego.shop/Customer/Profile

Function description:

Allows the customer to view and update their personal detail, including name, gender, email, phone, address, and date of birth.

Normal execution case:

Guest edits personal detail → clicks "Update" → system saves the changes.

Abnormal execution cases:

- Missing required fields.
- Invalid email or phone format.

Screen layout:

Profile picture, input fields for personal detail, gender selector, address field with location picker, date of birth picker, and "Update" button.

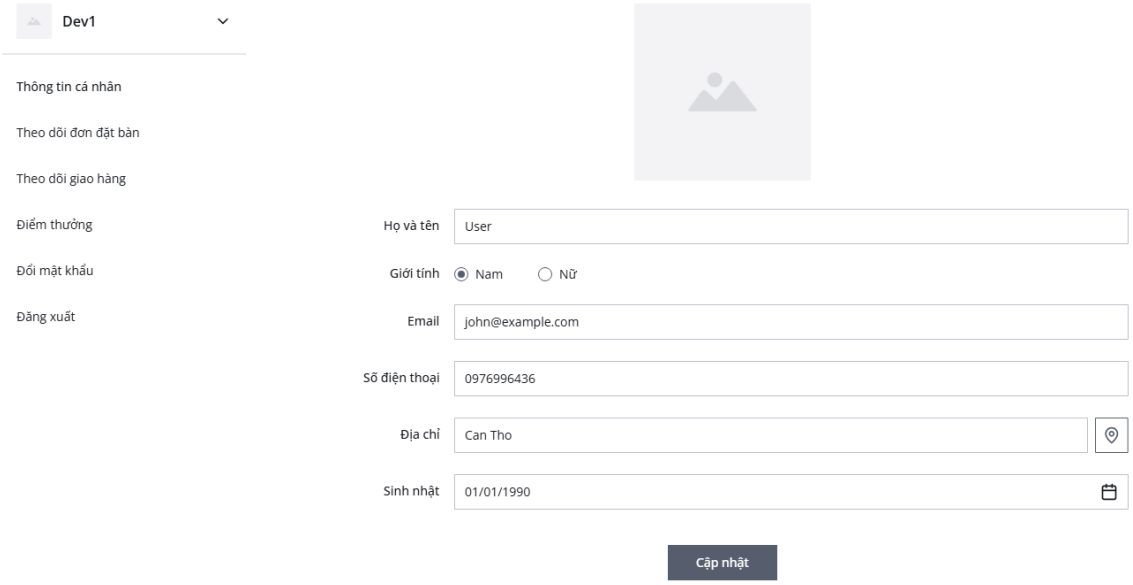


Figure 53: View/Edit Profile page of DineGO System

3.6.2 Change Password

Opened when the customer selects "Change Password" from their account menu. URL: https://dinego.shop/Customer/ChangePassword

Function description:

Allows the customer to create or update their account password by entering and confirming a new password.

Normal execution case:

Guest enters a new password and confirmation → clicks "Create Password" → system updates the password.

Abnormal execution cases:

- Missing password fields.
- Password and confirmation do not match.

Screen layout:

Password and confirm password input fields with visibility toggle, and a "Create Password" button.

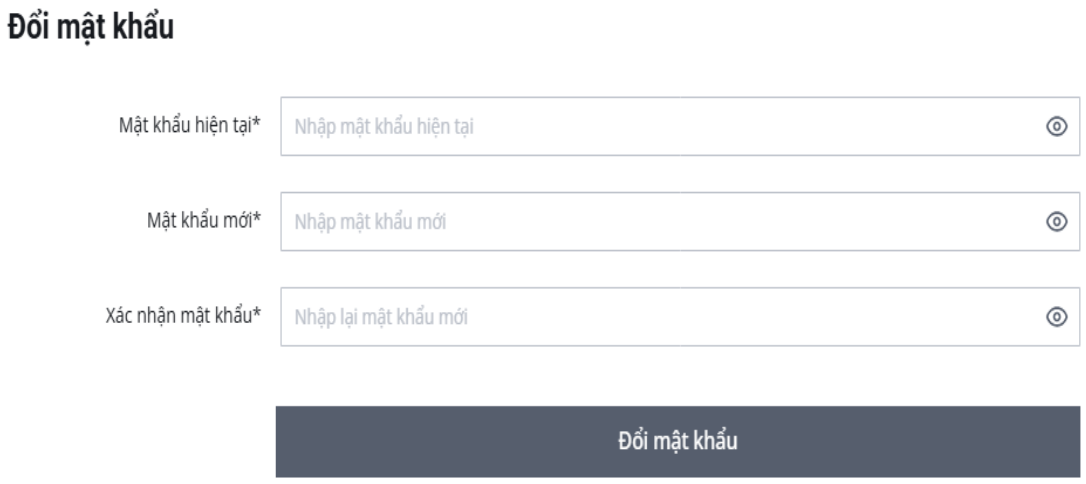


Figure 54: Change Password page of DineGO System

3.6.3 Apply as Owner

Function trigger:

Opened when the restaurant owner selects "Business License" from their dashboard menu. URL: https://dinego.shop/RestaurantOwner/Verifications

Function description:

Allows the restaurant owner to register or update business license information by entering license number, tax code, related documents, and uploading license files.

Normal execution case:

Owner fills in license detail → uploads license file → clicks "Register" → system saves the information.

Abnormal execution cases:

- Missing required fields.
- Invalid file format.

Screen layout:

Input fields for restaurant name, license number, tax code, related documents, file upload control, and "Register" button.

Figure 55: Apply as Owner page of DineGO System

3.7 Social Interaction

3.7.1 Posts

Function trigger:

Opened when the user navigates to the social feed page from the main menu. URL: https://dinego.shop/Post

Function description:

Displays posts from users, including text, images, and interactions such as like and comment. Allows users to share updates, like posts, and comment on them.

Normal execution case:

Guest scrolls through the feed → likes or comments on posts → optionally creates a new post.

Abnormal execution cases:

- No posts available.
- Failed to load posts.

Screen layout:

Profile section on the left, post creation box, and feed of posts with user name, timestamp, content, images, like count, and comment count.

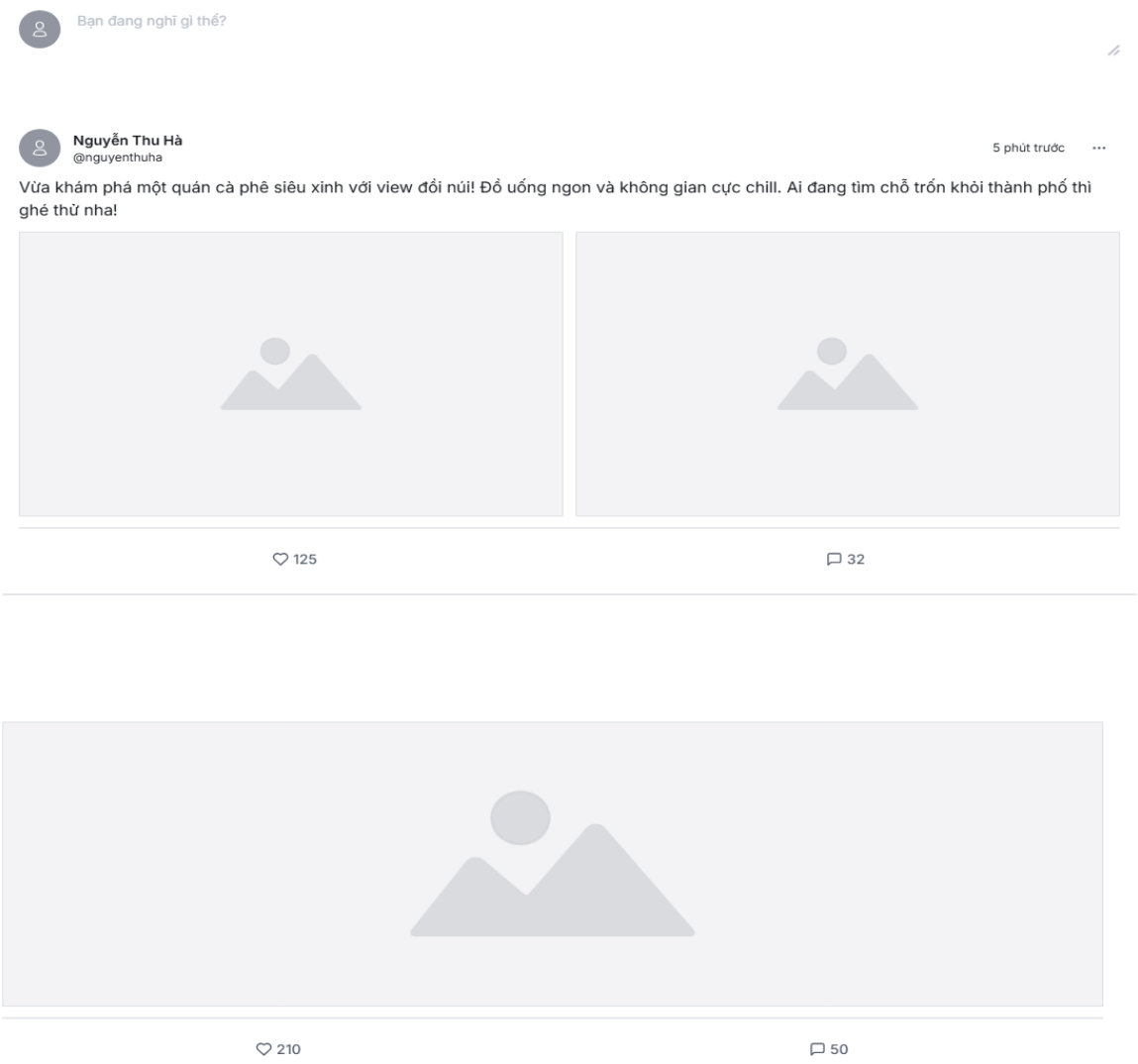


Figure 56: Posts page of DineGO System

3.7.2 Blogs

Function trigger:

Opened when the user navigates to the blog page from the main menu. URL: https://dinego.shop/Blog

Function description:

Displays popular blog posts of the month and newly published articles. Includes blog titles, images, publication dates, and related tags for filtering.

Normal execution case:

Guest browses the blog list → clicks on a post to read its content.

Abnormal execution cases:

- No blog posts available.
- Failed to load blog content.

Screen layout:
Popular blog section with images and titles, new blog section with thumbnails, titles, dates, and a tag list on the sidebar.

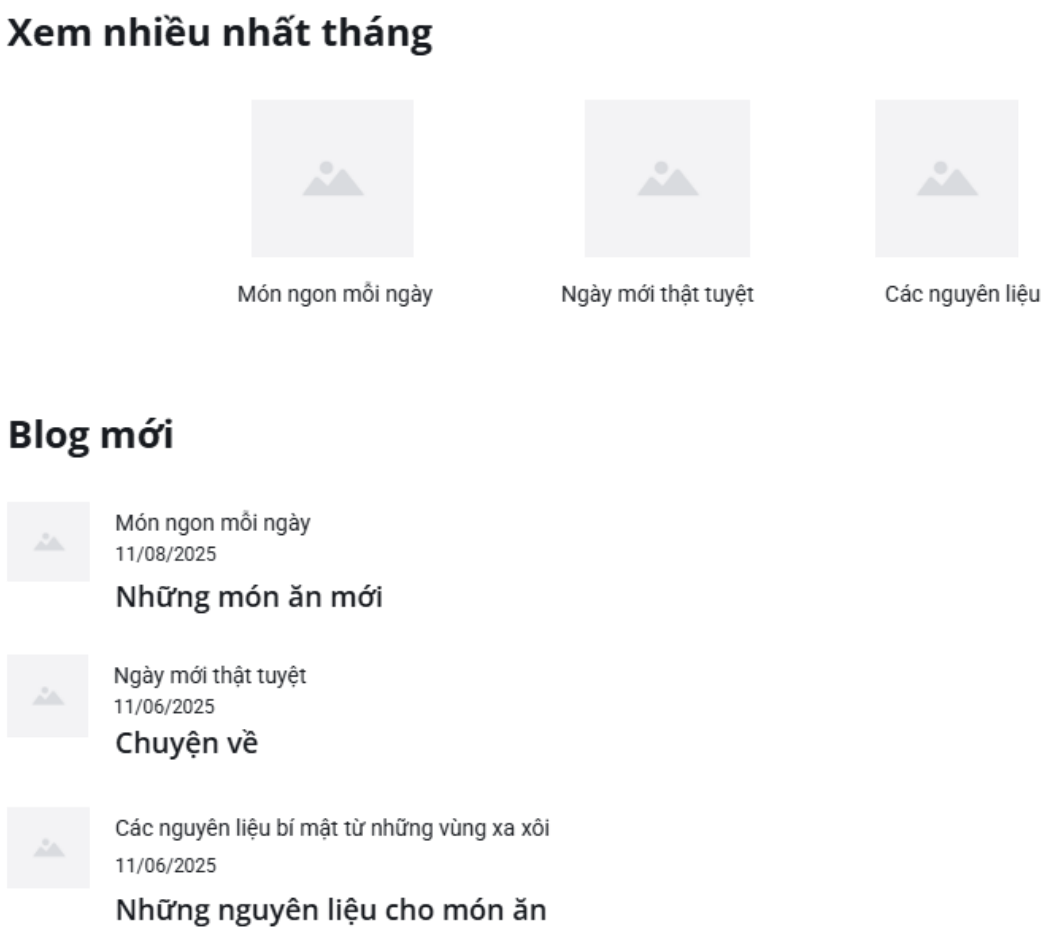


Figure 57: Blogs page of DineGO System

3.7.3 Post Detail

Function trigger:
Triggered when the user clicks on an image within a social post to view it in full size. URL: https://dinego.shop/Post

Function description:
Displays the selected image in an enlarged view along with post detail, likes, and comments. Allows navigation between images in the same post and interaction through likes or comments.

Normal execution case:
Guest clicks an image in a post → system opens image viewer → user can browse other images in the post, like, or comment.

Abnormal execution cases:
- Image not found.
- Failed to load post detail.

Screen layout:
Large image display area, post owner info, post caption, like count, comment section, and navigation arrows for multiple images.

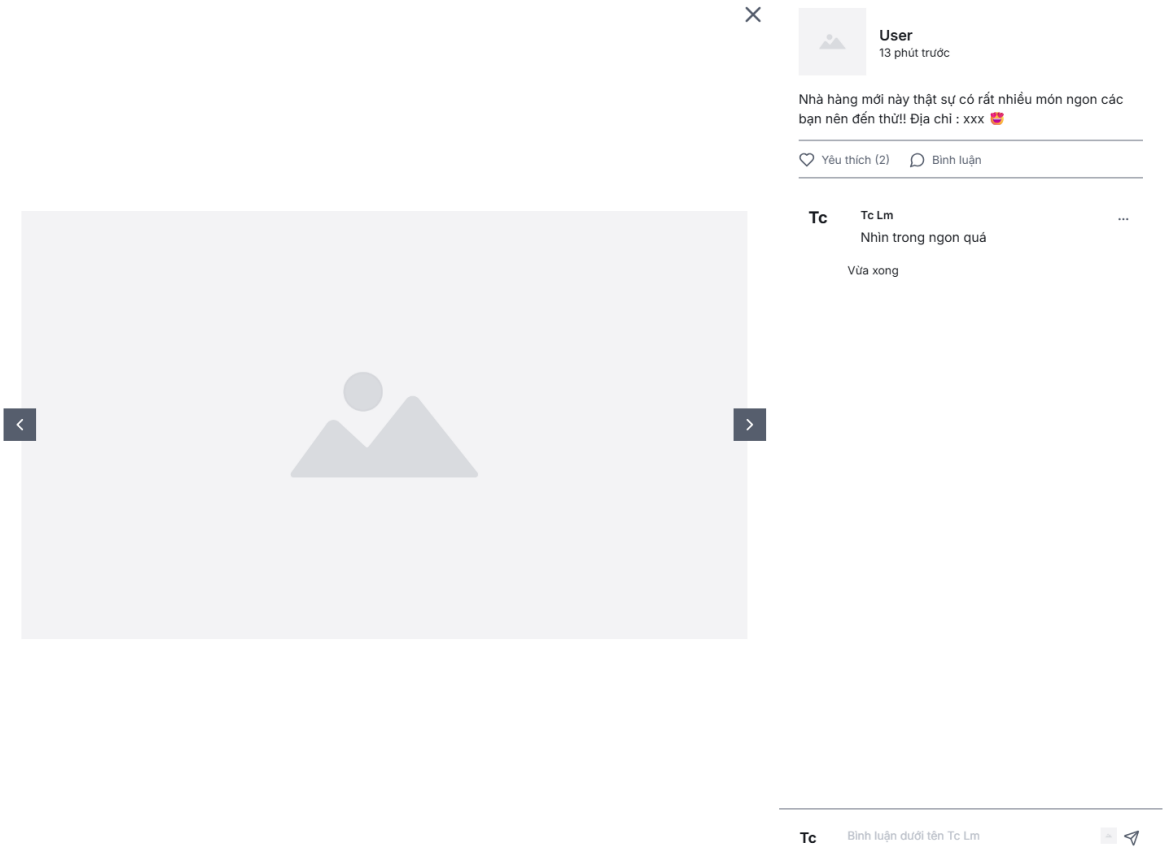


Figure 58: Post Detail page of DineGO System

3.7.4 Blog Detail

Function trigger:
Triggered when the user clicks on a blog post from the blog listing page. URL: https://dinego.shop/Blog/ViewBlogDetail/1

Function description:
Displays the full content of the selected blog post, including the title, publication date, and detailed article text. A sidebar shows links to other recent blog posts.

Normal execution case:
Guest clicks on a blog title → system loads the blog detail page → displays full content with related articles in the sidebar.

Abnormal execution cases:
- Blog post not found.
- Failed to load content.

Screen layout:
Main content area with blog title, date, and text content; sidebar showing thumbnails, titles, and dates of recent blogs.

Figure 59: Blog Detail page of DineGO System

3.7.5 Create Post

Function trigger:

Triggered when the user clicks the "Create Post" button on the social feed page. URL: https://dinego.shop/Post?showMine=True

Function description:

Allows the user to create a new social post by entering text and optionally adding images before publishing it to the feed.

Normal execution case:

Guest clicks label → enters text content → optionally uploads one or more images → clicks "Post" → system publishes the post to the feed.

Abnormal execution cases:

- Empty content without images → system prevents publishing and shows a warning.
- Failed to upload image.

Screen layout:

Profile avatar and name, text input area with placeholder, image upload button, emoji icon, and "Post" button.

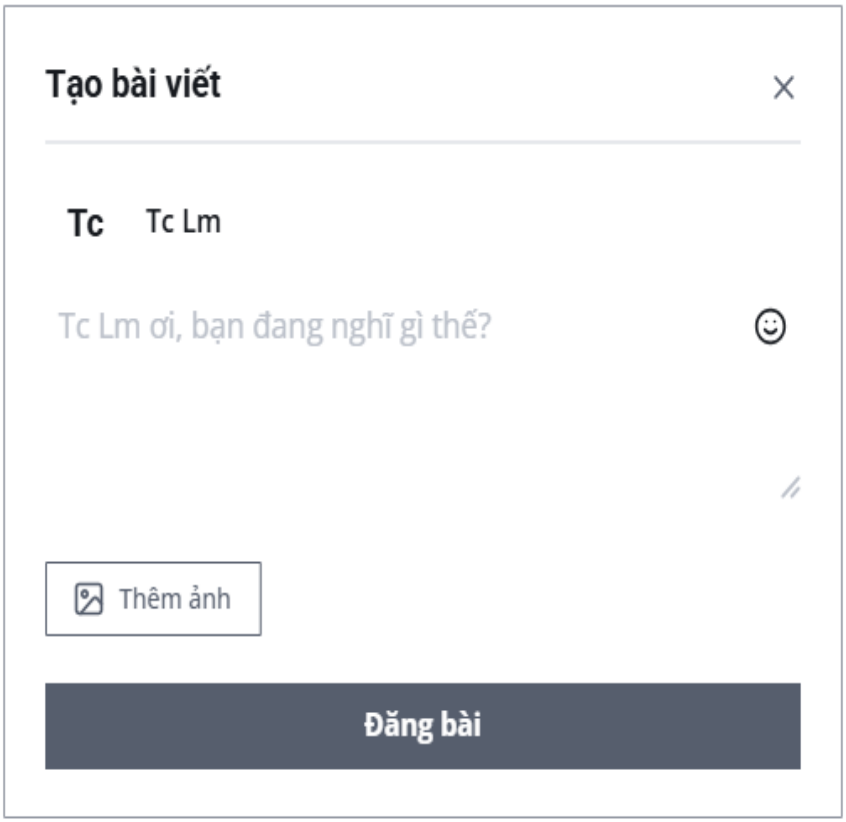


Figure 60: Create Post page of DineGO System

3.7.6 Report

Function trigger:

Triggered when the user navigates to the "Report Issue / Feedback" page from the main menu. URL: https://dinego.shop/Report

Function description:

Allows users to report issues or submit feedback to the system administrators by selecting a report type, entering detail, and optionally providing related links.

Normal execution case:

Guest opens the report page → selects a report type → enters description → optionally adds related link → clicks "Submit Report" → system sends the report to admins.

Abnormal execution cases:

- Missing report type → system prevents submission and shows an error.
- Missing description → system prevents submission and shows an error.
- Invalid link format → show warning.

Screen layout:

Dropdown for report type, text area for description, text input for related link, and "Submit Report" button.

Figure 61: Report page of DineGO System

3.7.7 Contact

Function trigger:

Triggered when the user navigates to the "Contact DineGO" page from the main menu. URL: https://dinego.shop/Contact

Function description:

Allows users to send feedback, inquiries, or support requests to DineGO administrators by filling in their name, email, subject, and message content.

Normal execution case:

Guest opens contact page → fills in all required fields → clicks "Send Contact" → system sends the message to the admin team for review and response.

Abnormal execution cases:

- Missing required fields (Name, Email, Subject, Content) → system prevents submission and shows error message.
- Invalid email format → system shows validation error.

Screen layout:

Input fields for full name, email, subject, message content, and "Send Contact" button.

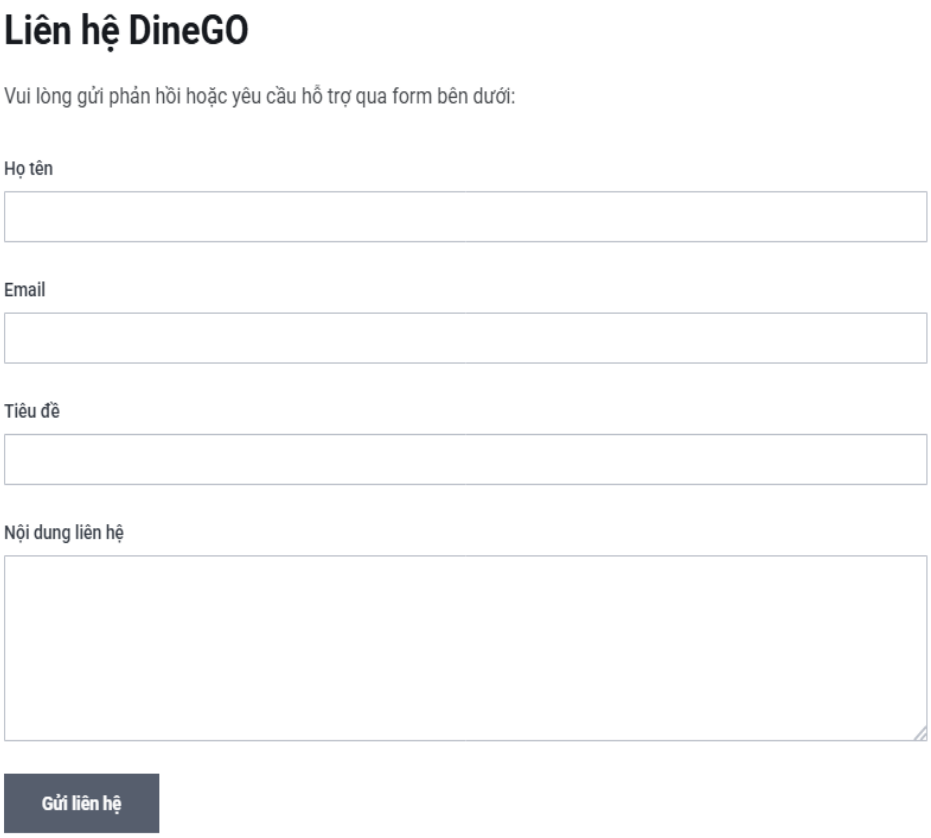


Figure 62: Contact page of DineGO System

3.7.8 Notification

Function trigger:

Triggered when the user clicks on the notification bell icon in the top navigation bar.

Function description:

Displays a popup with recent notifications for the logged-in user, such as system greetings, order updates, reservation changes, and other relevant alerts.

Normal execution case:

Guest clicks on the bell icon → system loads and displays the list of notifications (newest first) → user can switch between "All" and "Unread" tabs → clicking a notification redirects to the related page or action.

Abnormal execution cases:

- No notifications available → show “No notifications” message.
- Failed to load notifications → display an error message.

Screen layout:

Popup containing:

- Tab navigation for "All" and "Unread".
- List of notifications with icon, title, short message, timestamp, and link to related action.
- “View previous notifications” link at the bottom.

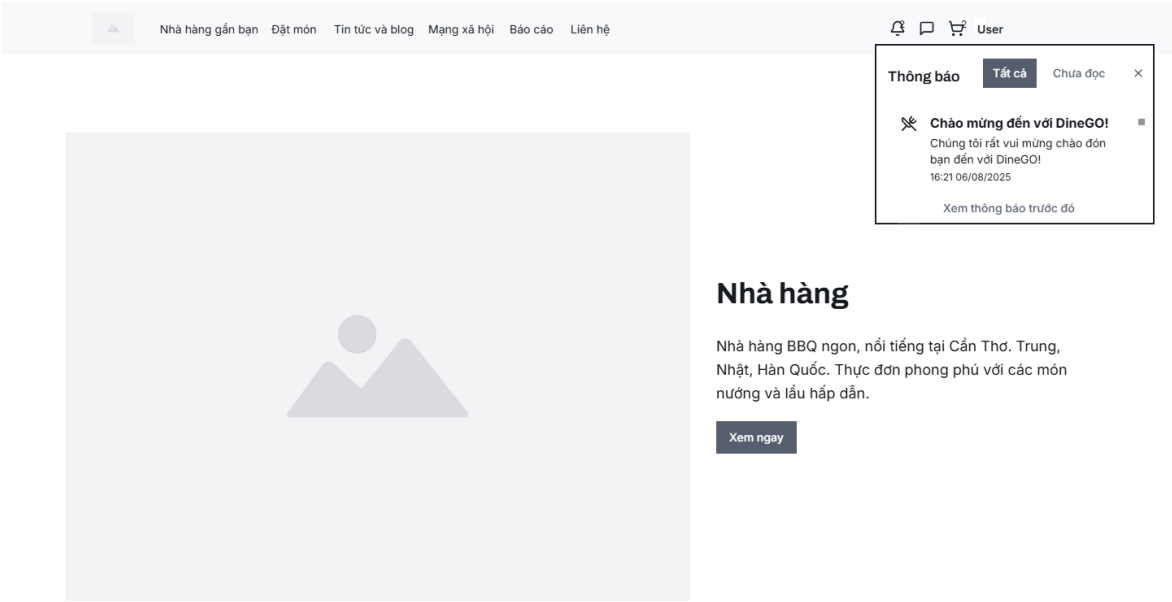


Figure 63: Notification of DineGO System

3.8 Communication & AI

3.8.1 Chat with Owner

3.8.1.1 Chat List Of Restaurant

Function trigger:

Click chat icon on navigation bar.

Function description:

Show pop-up with list of friends/restaurants to chat, search bar, and tabs to switch between lists.

Normal case:

Click icon → pop-up appears → select contact → open chat.

Abnormal case:

No contacts or search results → show message.

Screen layout:

Search bar, tabs, contact list with avatar and name.

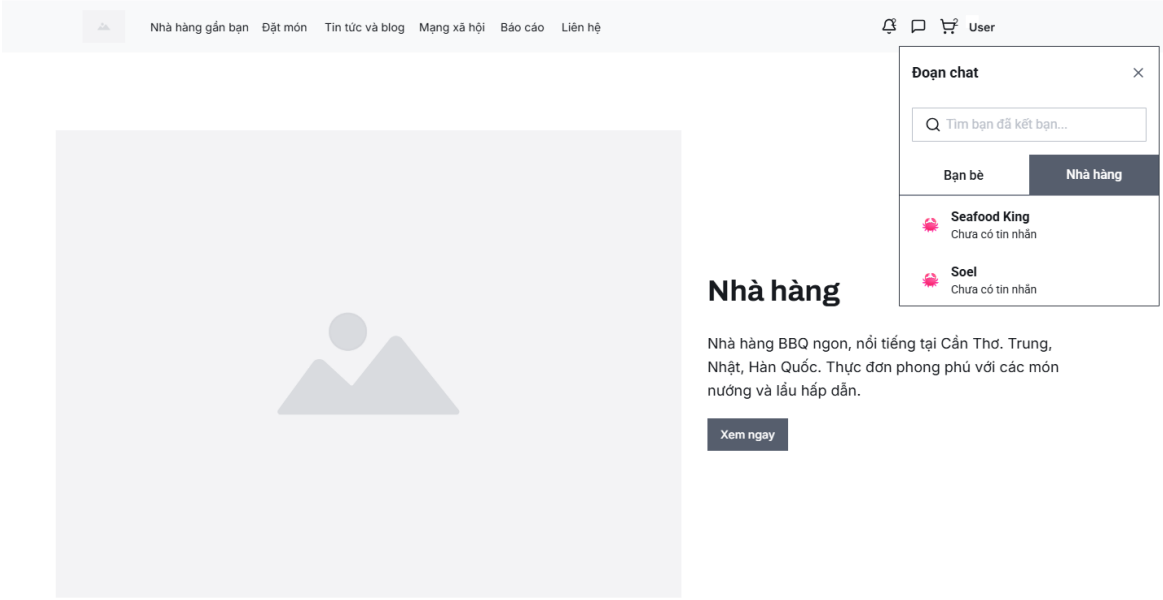


Figure 64: List Owner of DineGO System

3.8.1.2 Chat Box With Resowner

Function trigger:

Click “Chat” button on restaurant profile or support section.

Function description:

Opens a small chat window for real-time messaging with the restaurant.

Normal case:

Click button → pop-up chat opens → exchange messages in real-time.

Abnormal case:

Message send failure → show error.

Screen layout:

Header with restaurant name and close button, message history, input box, send button.

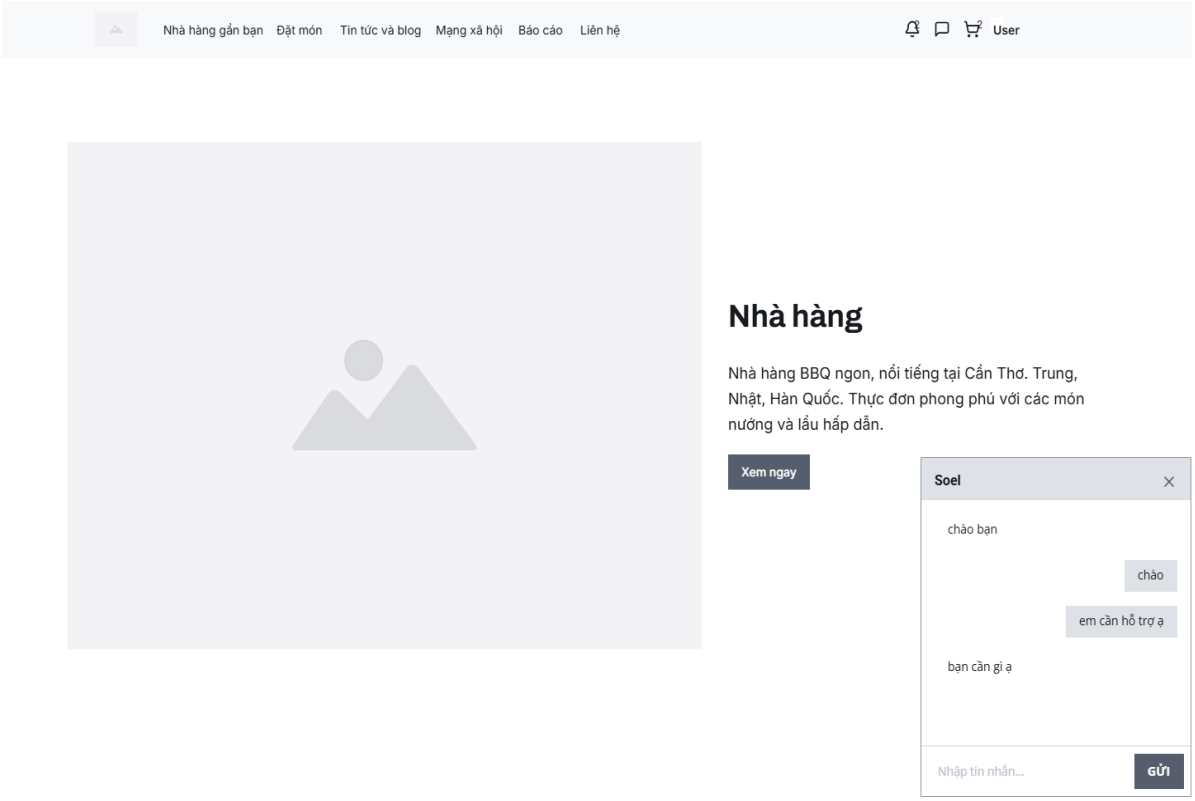


Figure 65: Chat with owner of DineGO System

3.8.2 Customer Chat With Customer

3.8.2.1 List Chat Customer With Customer

Function trigger:

Click chat icon on the navigation bar.

Function description:

Displays a pop-up list of friends or restaurants for quick chat access.

Normal case:

Click icon → pop-up opens with friend/restaurant tabs → select a contact → open chat window.

Abnormal case:

No contacts → show “No friends/restaurants” message.

Screen layout:

Search bar, tab buttons (Friends / Restaurants), contact list with name, avatar, and latest status.

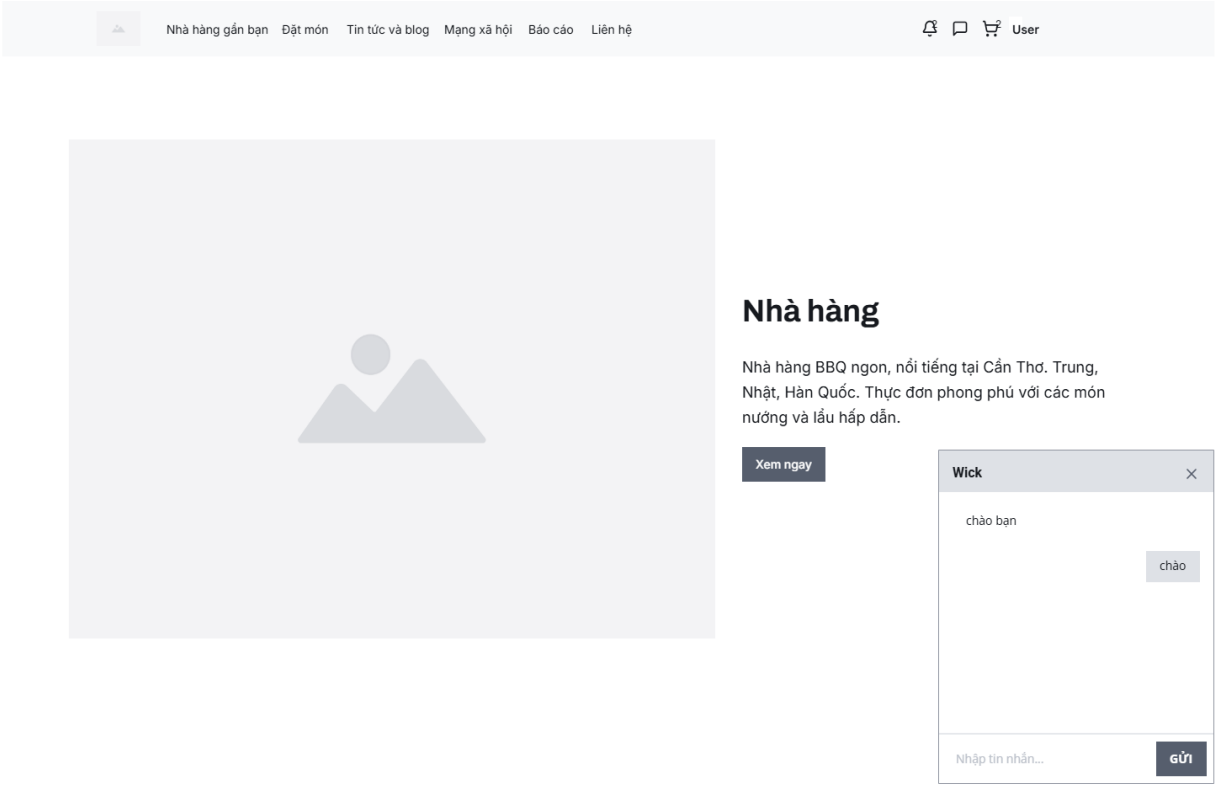


Figure 66: List Customer of DineGO System

3.8.2.2 Chat Box of Customer with Customer

Function trigger:

Click on a contact from the chat list pop-up.

Function description:

Opens a pop-up chat window for direct messaging with a friend or restaurant.

Normal case:

Click contact → pop-up opens → send and receive messages in real time.

Abnormal case:

Message sending fails → display error message.

Screen layout:

Header with contact name and status, message display area, text input box, and send button.

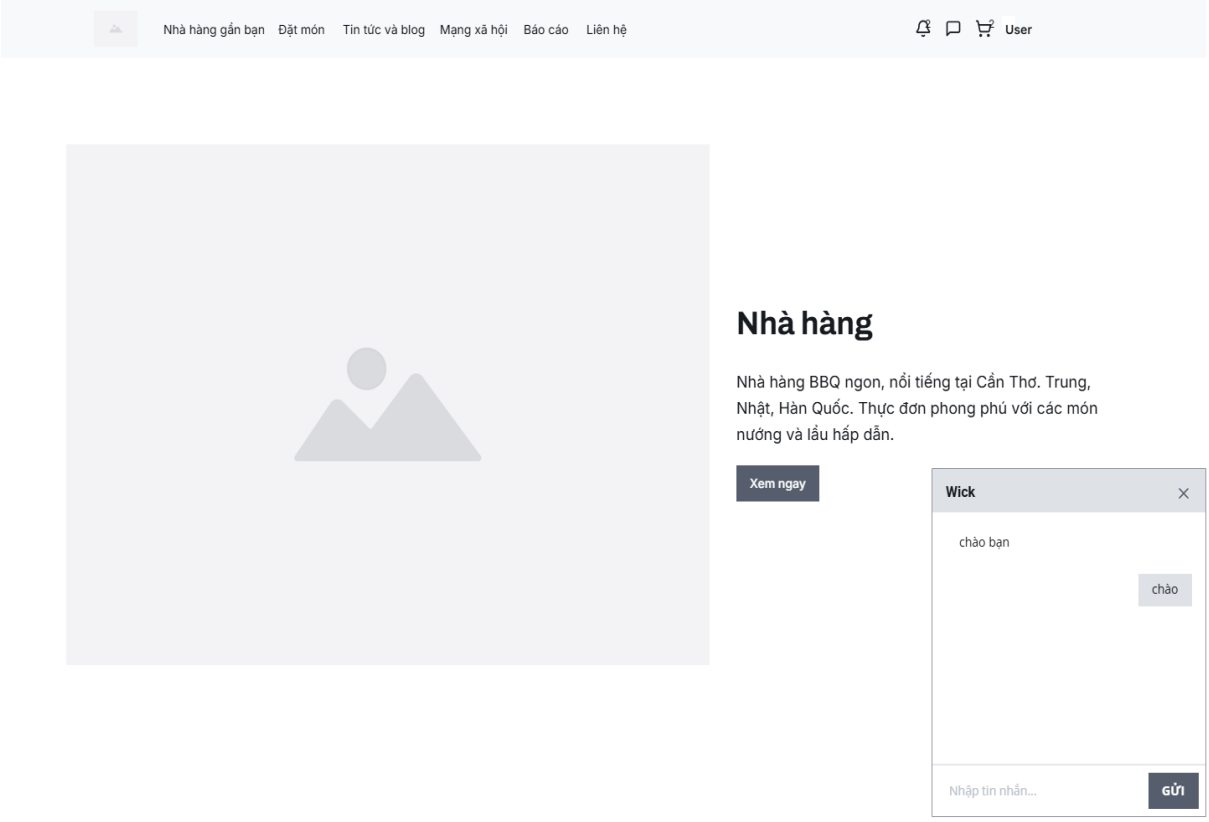


Figure 67: Customer chat box of DineGO System

3.8.3 AI Assistant

Function trigger:

Click the AI chat icon in the interface.

Function description:

Opens a pop-up chat window to interact with the AI assistant for instant responses.

Normal case:

Click AI icon → pop-up opens → user sends message → AI responds in real time.

Abnormal case:

AI service unavailable or request fails → system shows error messag**e.**

Screen layout:

Header with “AI Bot”, chat history, input box, send button.

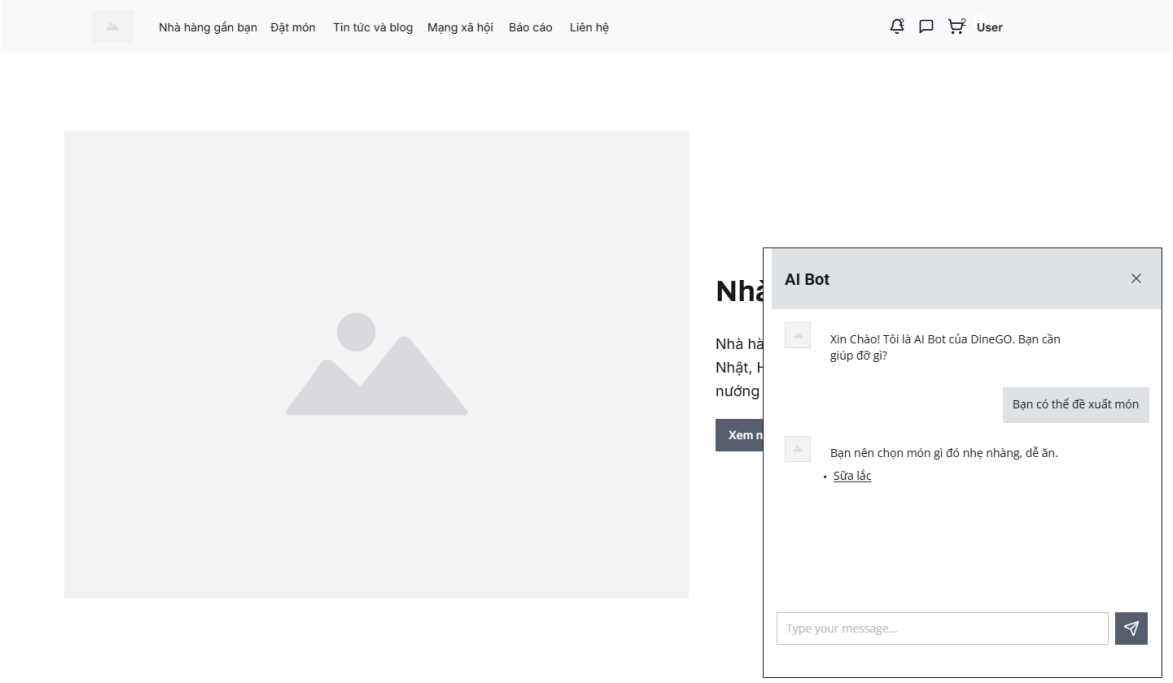


Figure 68: AI Assistant page of DineGO System

3.9 Owner Management

3.9.1 List Of Restaurant

Function trigger:

Accessed when the restaurant owner clicks **"Restaurant List"** from the side menu.

URL: https://dinego.shop/RestaurantOwner/ListRestaurant

Function description:

Displays a list of restaurants managed by the owner, each with an image, name, location, and quick action buttons. Allows adding new restaurants.

Normal execution case:

Owner opens "Restaurant List" → system displays all restaurants they manage → owner clicks “Access” to manage detail or “Add Restaurant” to create a new one.

Abnormal execution cases:

No restaurants found → system shows “No restaurants available” message.

Screen layout:

- Restaurant cards: image, name, city, “Access” button.
- “Add Restaurant” button for creating a new restaurant.

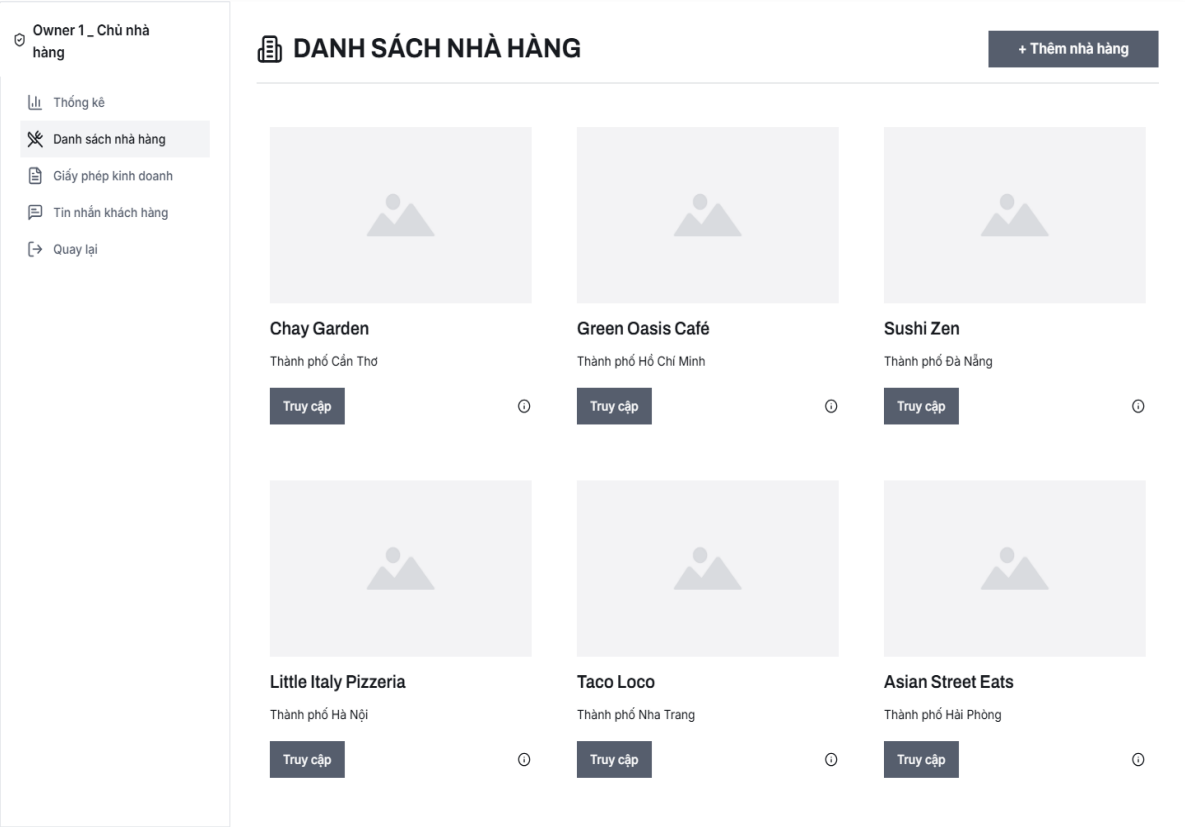


Figure 69: List of restaurant page of DineGO System

3.9.2 Chat With Customer

Function trigger:

Accessed when the restaurant owner clicks **"Customer Messages"** from the side menu.

URL: https://dinego.shop/RestaurantOwner/restaurantowner/messages

Function description:

Allows the restaurant owner to view and reply to messages from customers in real time. Displays a list of customer conversations and a chat area.

Normal execution case:

Owner opens "Customer Messages" → system displays a list of customers who have chatted → owner selects a customer → chat history loads → owner can type and send messages in real time.

Abnormal execution cases:

- No messages: show “No messages available” message.
- Customer not found: show “Customer not found” message when trying to open a conversation.
- Message send failure: show error if sending message fails (e.g., network error).

Screen layout:

- Left panel: list of customers with avatars, names, and message status (online/offline).
- Right panel: chat window showing message history and input box to send new messages.

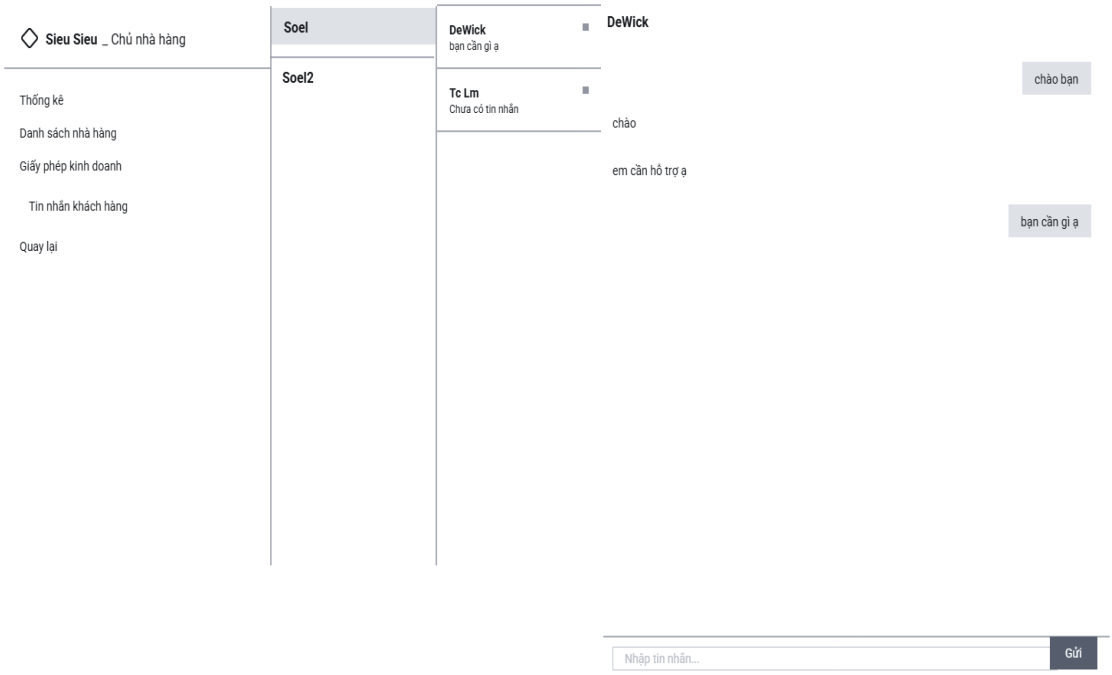


Figure 70: Chat with customer page of DineGO System

3.9.3 Manage Restaurant Info

3.9.3.1 Restaurant Owner Info

Function trigger: Accessed when the restaurant owner selects **"Restaurant Information"** from the side menu.

URL: https://dinego.shop/RestaurantOwner/ProfileRestaurant

Function description: Displays the current restaurant's detail including name, address, contact information, description, deposit fee, and promotions. Allows the owner to update the information.

Normal execution case: Owner opens "Restaurant Information" → system loads and displays current restaurant detail → owner clicks **"Update Information"** → system navigates to the edit page or shows the edit form.

Abnormal execution cases:

- Restaurant not found: Show "Restaurant not found" if the ID is invalid.
- Data load failure: Show error message if unable to fetch restaurant detail.
- Update failure: Show error if saving changes fails.

Screen layout:

- Left menu with restaurant management options.
- Main content area showing:
  - Restaurant image
  - Name, address, phone, email
  - Description
  - Deposit fee, promotions, and status (open/closed)
  - **"Update Information"** button.

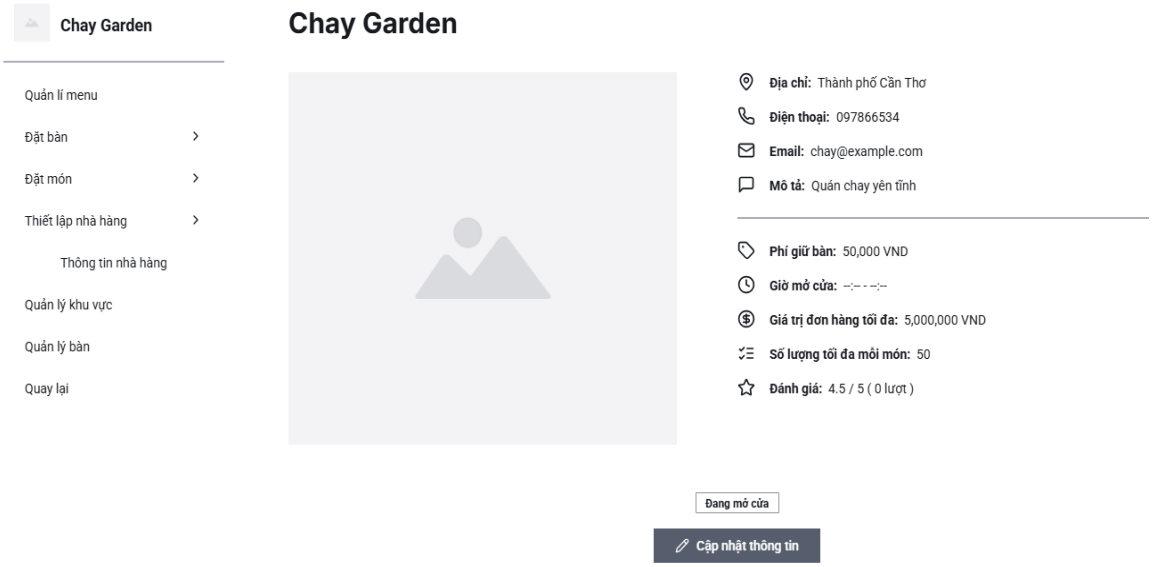


Figure 71: Restaurant Info page of DineGO System

3.9.3.2 Update Information

Function trigger:

Accessed when the restaurant owner clicks **"Update Information"** from the restaurant detail page.

URL: https://dinego.shop/RestaurantOwner/UpdateProfileRestaurant/5

Function description:

Allows the restaurant owner to edit and update restaurant information such as image, name, address, contact detail, description, deposit fee, promotions, and status.

Normal execution case:

Owner opens "Update Restaurant" form → edits the desired fields → clicks **"Save Changes"** → system validates input and updates the database → shows success message.

Abnormal execution cases:

- Missing required fields → show validation error.
- Invalid data format (e.g., non-numeric deposit fee) → show error message.
- Update fails due to server/database error → show failure notification.

Screen layout:

- Left menu with restaurant management options.
- Main form containing:
  - Image upload field (optional)
  - Restaurant name, address, email, phone number
  - Description
  - Deposit fee, promotions
  - Status toggle (active/inactive)
  - **"Save Changes"** and **"Back"** buttons.

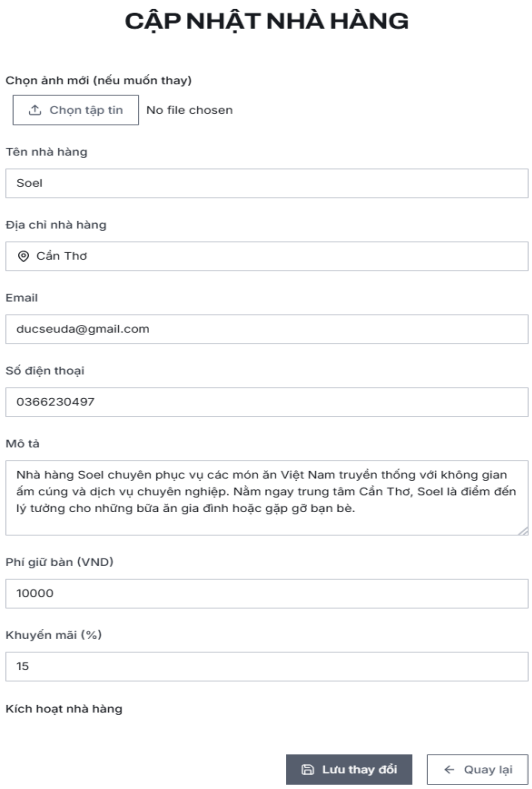


Figure 72: Restaurant edit Info page of DineGO System

3.9.4 Table Management

3.9.4.1 List Of table

Function trigger:

Accessed when the restaurant owner clicks **"Danh sách đặt bàn"** from the left menu under the "Đặt bàn" section.

URL: https://dinego.shop/Order

Function description:

Displays all tables in the restaurant with their current status, allowing the owner to quickly see which tables are available, booked, pending payment, or closed. The owner can filter tables by area and booking status.

Normal execution case:

Owner opens the table list page → selects **Area** and/or **Status** filters (optional) → system displays the matching tables with their detail (name, capacity, status color code).

Abnormal execution cases:

- No tables in the selected area → show "No tables found" message.
- Area or status filter not valid (e.g., deleted area) → reset to default view showing all tables.

Screen layout:

- **Filter section:**
  - Dropdown: Area selector
  - Dropdown: Status selector (All, Available, Pending Payment, Booked, Closed)
- **Table list section:**
  - Table cards showing:

- Table name
- Capacity
- Status color code (green = available, yellow = pending payment, red = booked, grey = closed).

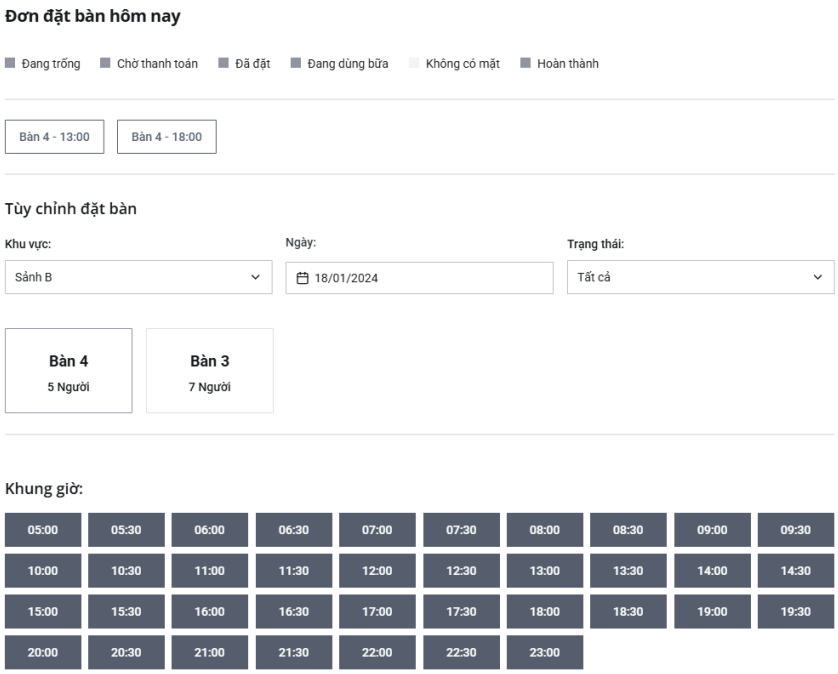


Figure 73: List of table page of DineGO System

3.9.4.2 Table list

Function trigger:

Triggered when the restaurant owner navigates to **"Manage Tables"** from the left-hand menu under **Restaurant Setup**.

URL: https://dinego.shop/Table/Index

Function description:

Enables the restaurant owner to view and manage all tables within a selected area. Owners can check table names, seating capacity, and create new tables if required.

Normal execution case:

Owner selects an **Area** from the dropdown → system displays all tables in that area with their names and capacities → owner clicks **Create Table** to add a new table.

Abnormal execution cases:

- No tables found in the selected area → display “No tables found” message.
- Selected area is invalid or deleted → reset area filter to default.

Screen layout:

- Dropdown: **Area Selector**.
- Button: **+ Create Table**.
- Table cards: Show table name and seating capacity.

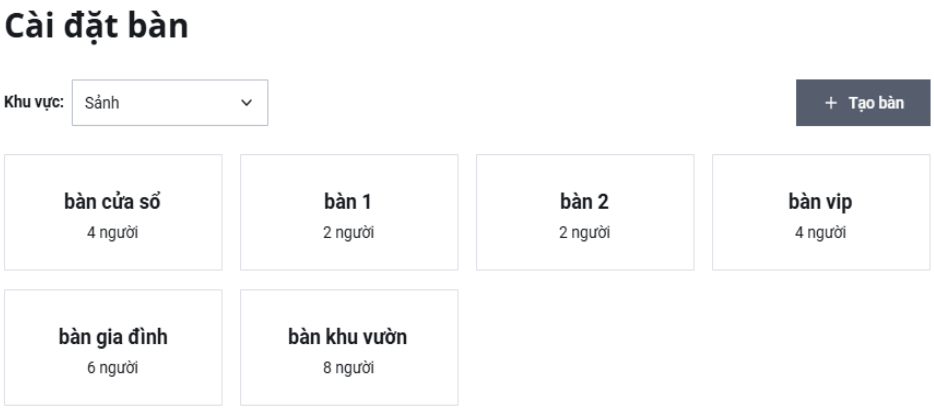


Figure 74: Table manager page of DineGO System

3.9.4.3 Create Table

Function trigger:

Triggered when the restaurant owner clicks **"+ Create Table"** on the **Manage Tables** page.

Function description:

Allows the restaurant owner to add a new table by specifying its name, number of seats, and optionally uploading a table image.

Normal execution case:

Owner enters **Table Name** → selects/enters **Number of Seats** → uploads an image (optional) → clicks **Add Table** → system saves and displays the new table in the list.

Abnormal execution cases:

- **Missing table name** → show validation message "Please enter table name".
- **Number of seats invalid** (e.g., less than 1) → show validation message "Number of seats must be at least 1".
- **Upload fails** → show error and allow retry.

Screen layout:

- Field: **Table Name** (input text).
- Field: **Number of Seats** (numeric input).
- Button: **Add Image** (optional).
- Button: **Add Table** (save).
- Close icon (top right) to cancel.

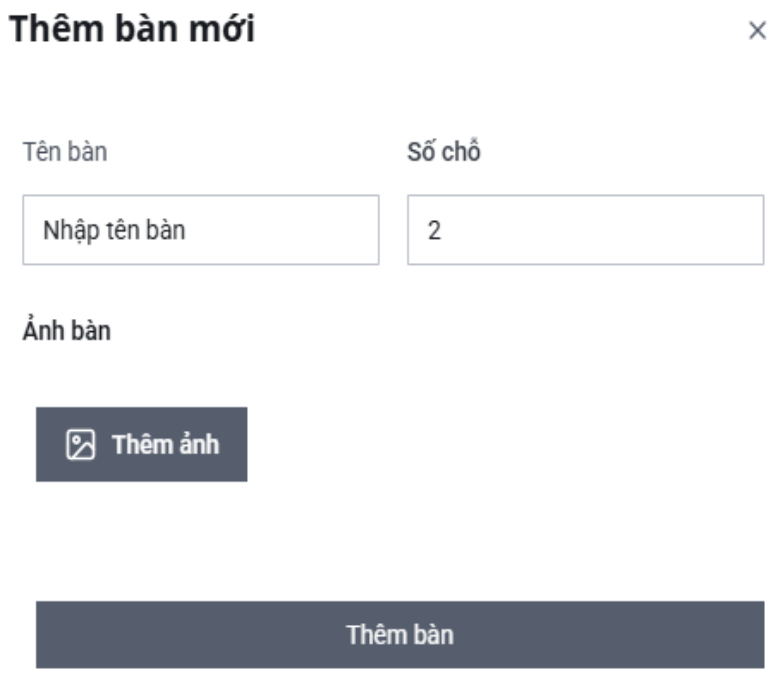


Figure 75: Table Create manager page of DineGO System

3.9.4.4 Edit Table

Function trigger:

Triggered when the restaurant owner clicks **"+ Create Table"** on the **Manage Tables** page.

Function description:

Allows the restaurant owner to add a new table by specifying its name, number of seats, and optionally uploading a table image.

Normal execution case:

Owner enters **Table Name** → selects/enters **Number of Seats** → uploads an image (optional) → clicks **Add Table** → system saves and displays the new table in the list.

Abnormal execution cases:

- **Missing table name** → show validation message “Please enter table name”.
- **Number of seats invalid** (e.g., less than 1) → show validation message “Number of seats must be at least 1”.
- **Upload fails** → show error and allow retry.

Screen layout:

- Field: **Table Name** (input text).
- Field: **Number of Seats** (numeric input).

- Button: **Add Image** (optional).
- Button: **Add Table** (save).

- Close icon (top right) to cancel.

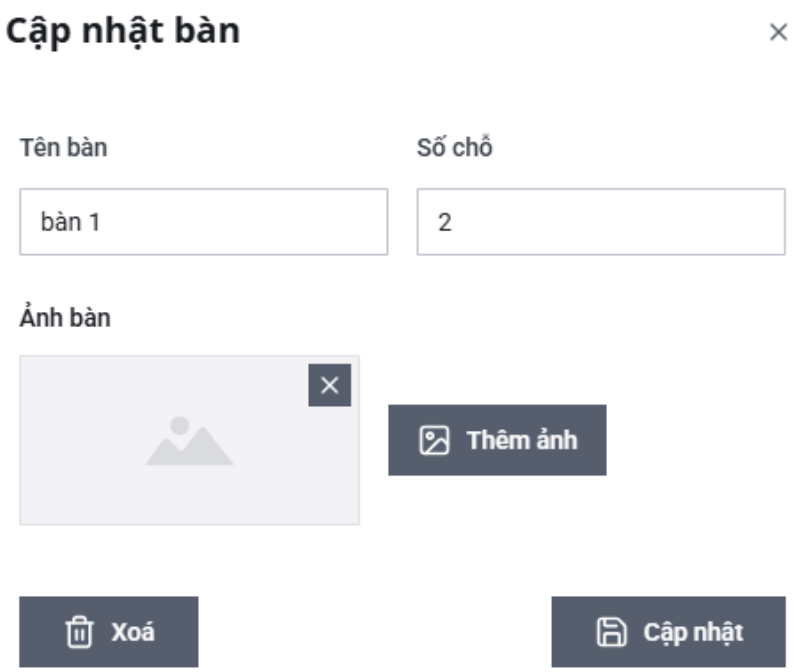


Figure 76: Table edit manager page of DineGO System

3.9.4.5 Delete table

Function trigger:

Triggered when the restaurant owner clicks the **Delete** button in the Edit Table popup.

Function description:

Displays a confirmation popup asking the restaurant owner to confirm deletion of the selected table. If confirmed, the system removes the table from the restaurant's table list.

Normal execution case:

- Owner clicks **Delete** → confirmation popup appears.
- Owner clicks **Delete** in the popup → system deletes the table and updates the list view.
- Owner clicks **Cancel** → popup closes, no changes are made.

Abnormal execution cases:

- Delete request fails (e.g., network error, server error) → show error message "Unable to delete table. Please try again."
- Table no longer exists (deleted by another session) → show message "Table not found." and refresh list.

Screen layout – Delete Table Confirmation popup:

- **Title:** "Delete Table Confirmation" (Xác nhận xóa bàn)
- **Message:** "Are you sure you want to delete table [Table Name]?" (Bạn có chắc muốn xóa bàn [Tên bàn] không?)
- **Buttons:**
  - **Cancel** (close popup, no action)
  - **Delete** (confirm deletion)

- **Close icon** (top right) to dismiss popup without changes

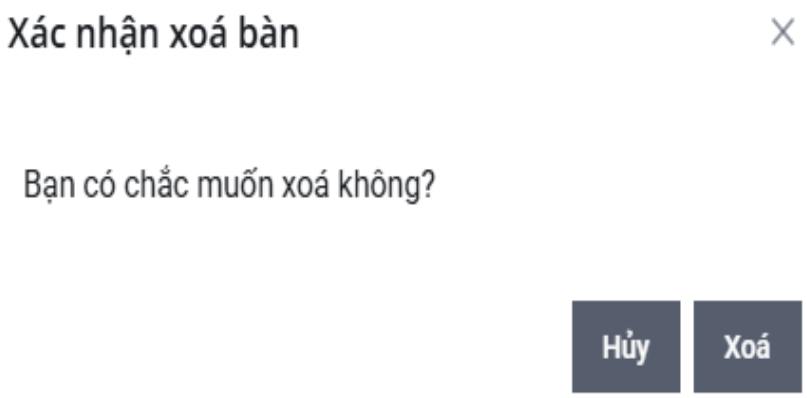


Figure 77: Table delete manager page of DineGO System

3.9.5 View Table Order List.

Function trigger:

Triggered when the restaurant owner navigates to **Restaurant Setup → Manage Tables** from the left menu.

URL:https://dinego.shop/Table/ReservationHistory

Function description:

Allows the restaurant owner to view and manage all tables within a selected area. Owners can check table names and seating capacity, filter by area, and create new tables.

Normal execution case:

Owner selects an **Area** from the dropdown → system loads and displays all tables in that area (name, capacity) → owner can click **+ Create Table** to add a new table.

Abnormal execution cases:

- No tables found in the selected area → show message: "No tables found".
- Selected area is invalid or deleted → reset area filter to default and reload list.
- Data load error → show a generic error and a **Retry** action.

Screen layout:

- **Dropdown:** Area Selector
- **Button:** + Create Table
- **Table cards/list:** Each item shows Table Name and Seating Capacity (optional thumbnail image)
- **Empty state:** "No tables found" with a hint to create a table

Figure 78: Table order list page of DineGO System

3.9.6 Area Manager

3.9.6.1 Area List

Function trigger:

Triggered when the restaurant owner navigates to **Restaurant Setup → Area Manager** from the left menu.

URL:https://dinego.shop/Table/ViewArea

Function description:

Allows the restaurant owner to view, create, update, and delete areas in the restaurant. Each area can be used to group tables for easier management.

Normal execution case:

Owner views a list of existing areas with their names and creation dates → can click **Create Area** to add a new area → can use **Edit** or **Delete** buttons to manage existing areas.

Abnormal execution cases:

- No areas found → display "No areas found" message.
- Area creation/update fails → display error message and allow retry.
- Attempt to delete an area that still has tables → show warning and prevent deletion.

Screen layout:

- **Button:** + Create Area (top right)
- **Table/List:** Columns for ID, Area Name, Creation Date, and Actions (Edit, Delete)
- **Empty state:** Informative message with a link to create a new area

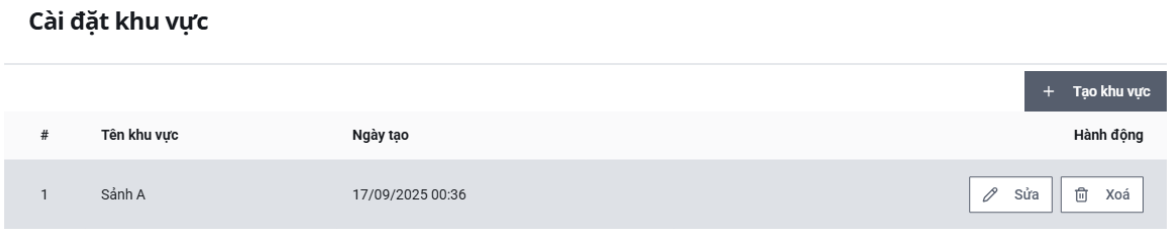


Figure 79: Area list page of DineGO System

3.9.6.2 Create Area

Function trigger:
Triggered when the restaurant owner clicks **+ Create Area** in the Area Manager page.
Function description:
Displays a popup form allowing the owner to enter a new area name and save it to the system.
Normal execution case:
Owner enters the area name → clicks **Save** → system creates the area and updates the list.
Abnormal execution cases:

- Area name is empty → show validation message.
- Save fails → show error message.

Screen layout:
Popup with:

- **Field:** Name (text input
- **Button:** Save
- **Close icon:** to cancel

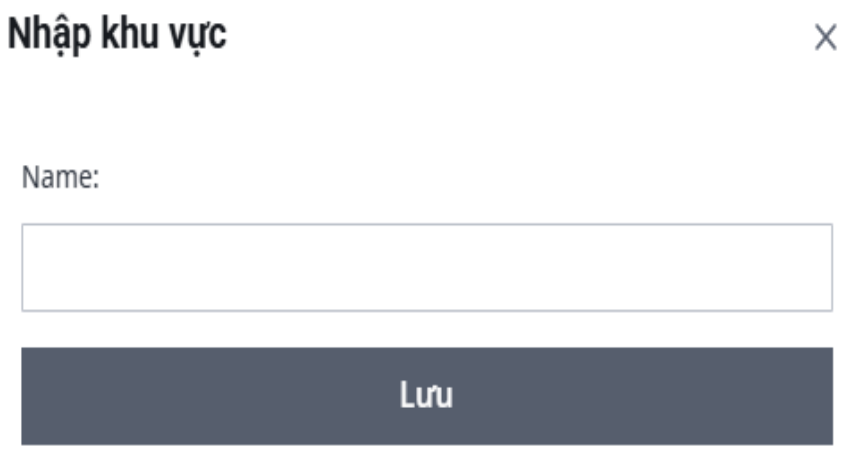


Figure 80: Area create page of DineGO System

3.9.6.3 Edit Area

Function trigger:
Triggered when the restaurant owner clicks **Edit** on an area in the Area Manager page.
Function description:
Displays a popup form allowing the owner to modify the selected area name and save changes.
Normal execution case:
Owner updates the area name → clicks **Save** → system updates the area and refreshes the list.
Abnormal execution cases:

- Area name is empty → show validation message.
- Save fails → show error message.

Screen layout:
Popup with:

- **Field:** Area Name (text input, pre-filled)
- **Button:** Save

- **Close icon:** to cancel

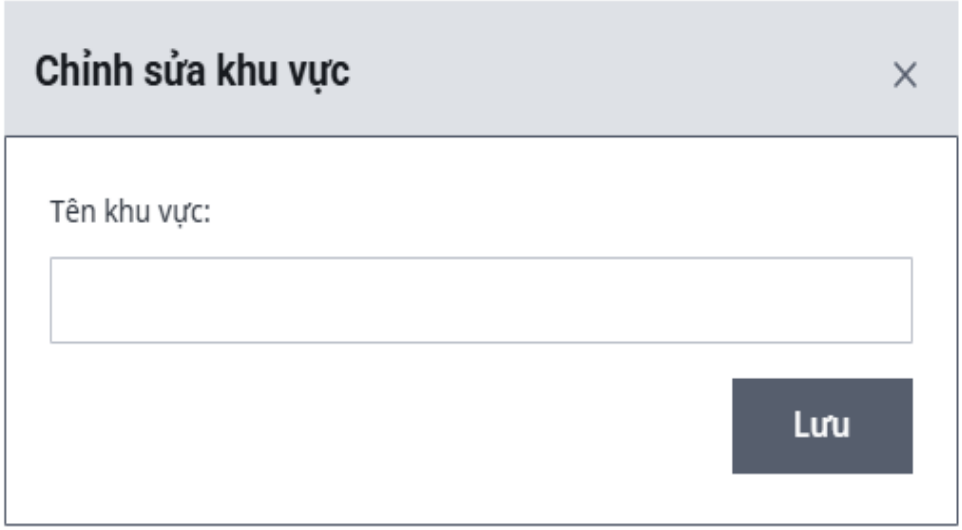


Figure 81: Area edit page of DineGO System

3.9.6.4 Delete Area

Function trigger:

Triggered when the restaurant owner clicks **Delete** on an area in the Area Manager page.

Function description:

Displays a confirmation popup asking the owner to confirm before deleting the selected area.

Normal execution case:

Owner clicks **Delete** → system removes the area → updates the list.

Abnormal execution cases:

- Area deletion fails → show error message.
- Area is in use (has tables) → show warning and prevent deletion.

Screen layout:

Popup with:

- Confirmation message (area name highlighted)
- **Buttons:** Cancel, Delete
- **Close icon:** to cancel

Figure 82: Area delete page of DineGO System

3.9.7 Menu & Food CRUD

3.9.7.1 Menu List

Function trigger:

Triggered when the restaurant owner navigates to **Menu Management** from the left menu.

URL: https://dinego.shop/Menu/ListMenu/5

Function description:

Allows the restaurant owner to view all menus of the selected restaurant area, including menu name, type, and image. Owners can create, edit, delete menus, or manage the foods inside a menu.

Normal execution case:

Owner sees a list of menus → can click **Add Menu** to create new → can click **Edit**, **Delete**, or **Manage Foods** for an existing menu.

Abnormal execution cases:

- No menus found → display "No menus found" message.
- Action fails (create, edit, delete) → show error message.

Screen layout:

- **Button:** Add Menu (top right)
- **Table/List:** Columns for Menu Name, Type, Image, and Actions (Edit, Delete, Manage Foods)
- **Empty state:** Message with a link to add a new menu

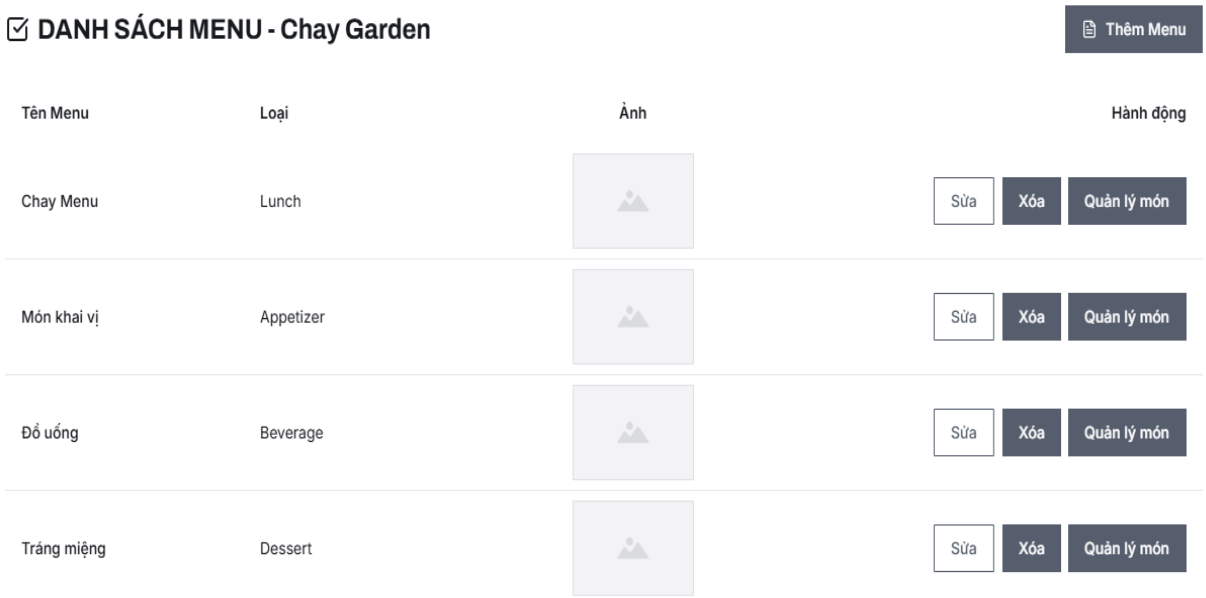


Figure 83: Menu list page of DineGO System

3.9.7.2 Create Menu

Function trigger:

Triggered when the restaurant owner clicks **Add Menu** on the Menu List page.

URL: https://dinego.shop/Menu/Create/5

Function description:

Allows the owner to create a new menu by entering menu name, selecting menu type, and optionally uploading an image.

Normal execution case:

Owner fills in all required fields → clicks **Add** → system saves the new menu and updates the list.

Abnormal execution cases:

- Missing required fields → show validation message.
- Save fails → show error message.

Screen layout:

- **Fields:** Menu Name (text), Menu Type (text), Menu Image (file upload)
- **Buttons:** Back, Add

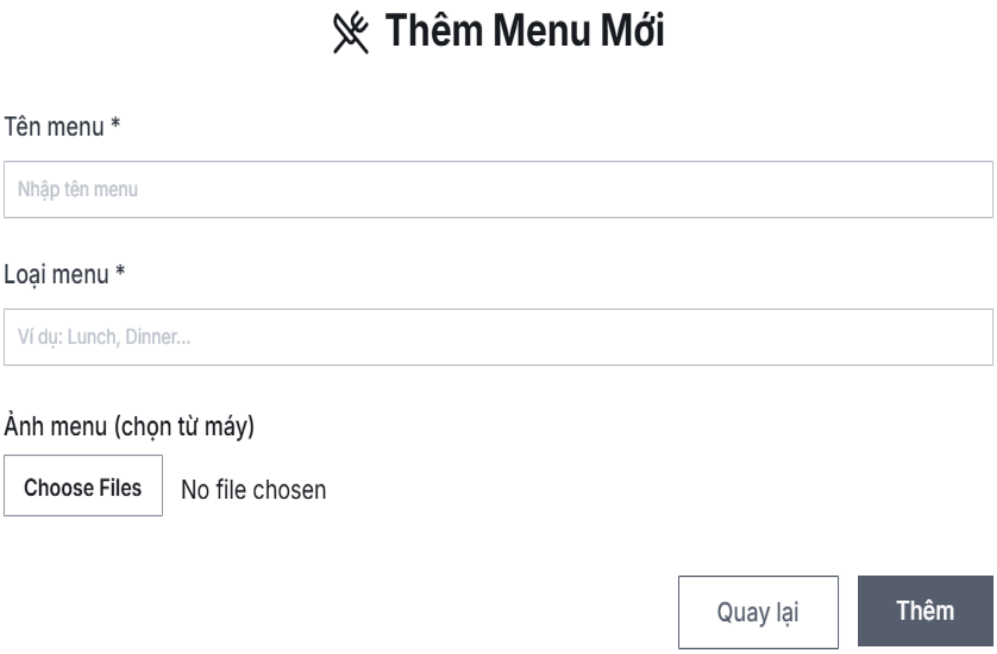


Figure 84: Menu Create page of DineGO System

3.9.7.3 Edit Menu

Function trigger:

Triggered when the restaurant owner clicks **Edit** on a menu in the Menu List page.

URL: https://dinego.shop/Menu/Edit/4

Function description:

Allows the owner to update menu detail such as name, type, and optionally upload a new image.

Normal execution case:

Owner updates the required fields → clicks **Save** → system saves changes and updates the list.

Abnormal execution cases:

- Missing required fields → show validation message.
- Save fails → show error message.

Screen layout:

- **Fields:** Menu Name (text), Menu Type (text), New Image (file upload)
- **Buttons:** Back, Save

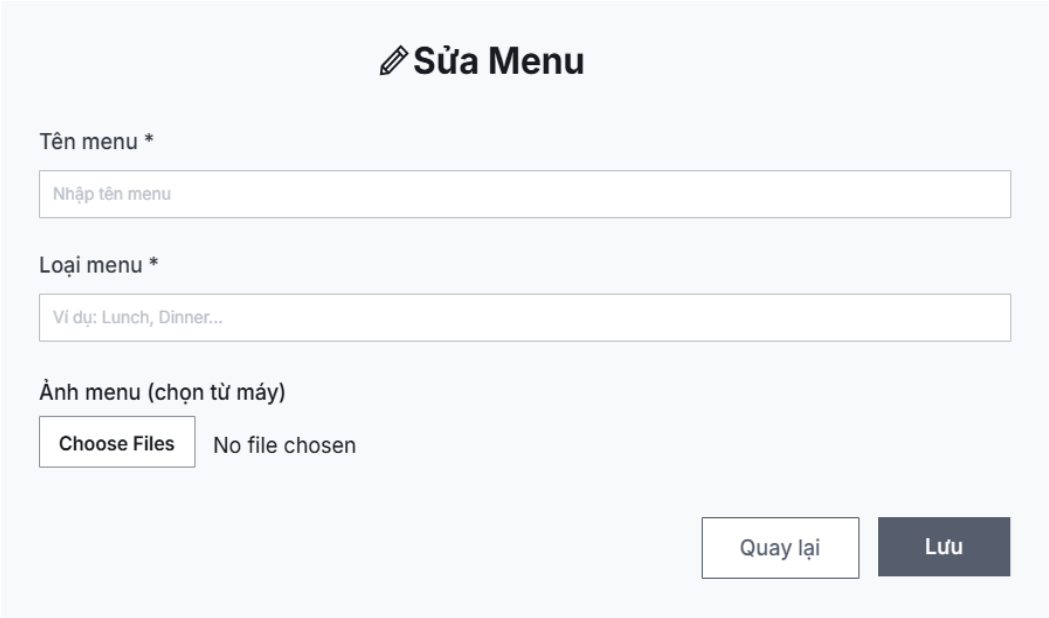


Figure 85: Menu edit page of DineGO System

3.9.7.4 Delete Menu

Function trigger:

Triggered when the restaurant owner clicks **Delete** on a menu in the Menu List page.

URL: https://dinego.shop/Menu/Delete/4

Function description:

Displays a confirmation message before deleting the selected menu.

Normal execution case:

Owner confirms deletion → system removes the menu → updates the list.

Abnormal execution cases:

- Delete fails → show error message.
- Menu is in use (has linked foods/orders) → show warning and prevent deletion.

Screen layout:

- Confirmation message (menu name highlighted)
- **Buttons:** Delete, Cancel

Figure 86: Menu delete page of DineGO System

3.9.7.5 Food List

Function trigger:

Triggered when the restaurant owner clicks **Manage Foods** on a menu in the Menu List page.

URL: https://dinego.shop/Food/ManageByMenu?menuId=4

Function description:

Displays a list of foods in the selected menu, including name, description, price, image, and status. Owners can add, edit, or delete foods.

Normal execution case:

Owner views list → can click **Add Food**, **Edit**, or **Delete** for any item.

Abnormal execution cases:

- No foods found → display "No foods found" message.
- Action fails (create, edit, delete) → show error message.

Screen layout:

- **Button:** Add Food (top right)
- **Table/List:** Columns for Name, Description, Price, Image, Status, and Actions (Edit, Delete)
- **Empty state:** Message with link to add a new food

Figure 87: Food list page of DineGO System

3.9.7.6 Create Food

Function trigger:

Triggered when the restaurant owner clicks **Edit** on a food item in the Food List page.

URL: https://dinego.shop/Create/4

Function description:

Allows the owner to update food detail such as name, description, tags, price, status, and image.

Normal execution case:

Owner updates necessary fields → clicks **Update** → system saves changes and updates the list.

Abnormal execution cases:

- Missing required fields → show validation message.
- Save fails → show error message.

Screen layout:

- **Fields:** Food Name (text), Description (text area), Tags (multi-select), Price (numeric), Status (dropdown), Current Image, New Image (file upload)
- **Buttons:** Back, Update

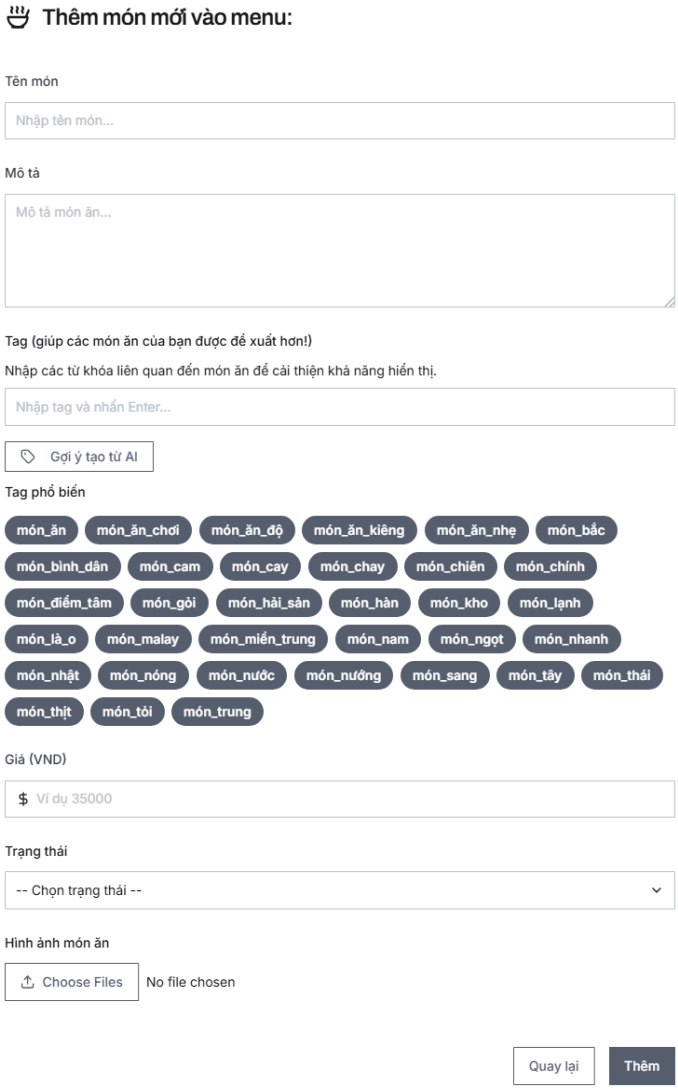


Figure 88: Food create page of DineGO System

3.9.7.7 Edit Food

Function trigger:
Triggered when the restaurant owner clicks **Edit** on a food item in the Food List page.
URL: https://dinego.shop/Edit/4
Function description:
Allows the owner to update food information such as name, description, tags, price, status, and image.
Normal execution case:
Owner edits the desired fields → clicks **Update** → system saves changes and refreshes the list.
Abnormal execution cases:
Missing required fields → show validation message.
Save fails → show error message.
Screen layout:
Fields: Food Name, Description, Tags (multi-select), Price, Status, Current Image, New Image upload
Buttons: Back, Update

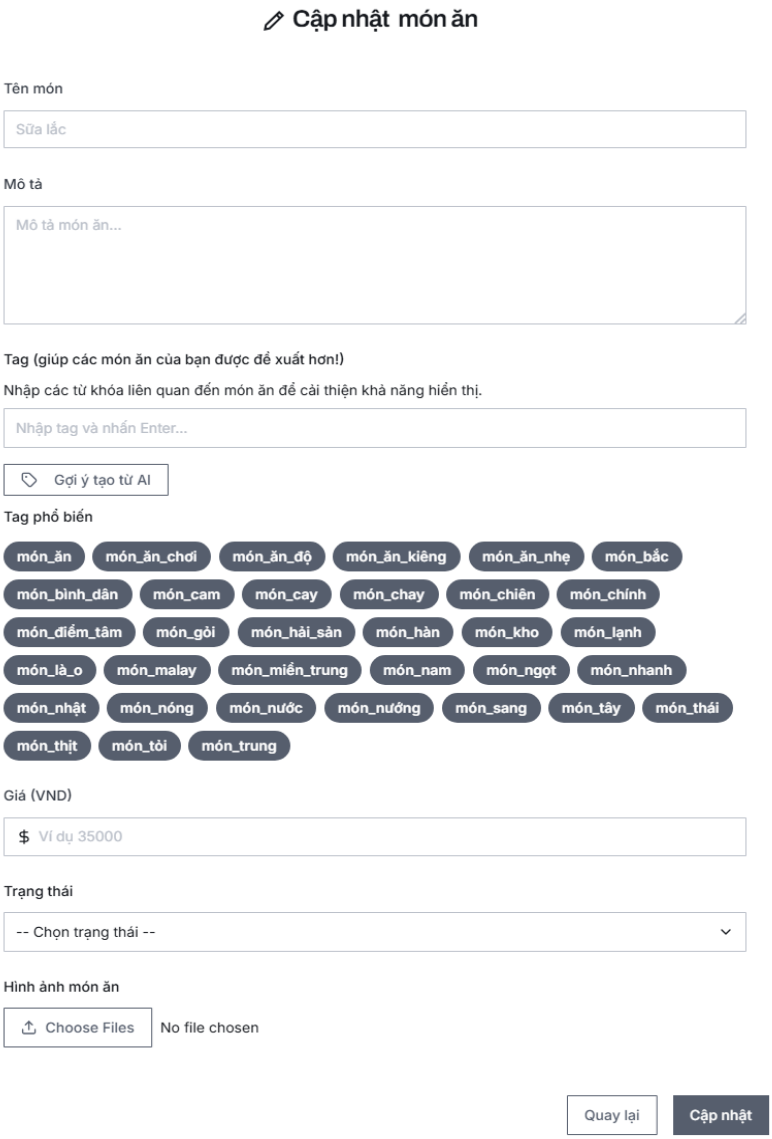


Figure 89: Food edit page of DineGO System

3.9.7.8 Delete Food

Function trigger:

Triggered when the restaurant owner clicks **Delete** on a food item in the Food List page.

URL: https://dinego.shop/Delete/4

Function description:

Displays a confirmation message before deleting the selected food.

Normal execution case:

Owner confirms deletion → system removes the food → updates the list.

Abnormal execution cases:

- Delete fails → show error message.
- Food is in use (linked to orders) → show warning and prevent deletion.

Screen layout:

- Confirmation message (food name highlighted)

- **Buttons:** Delete, Cancel

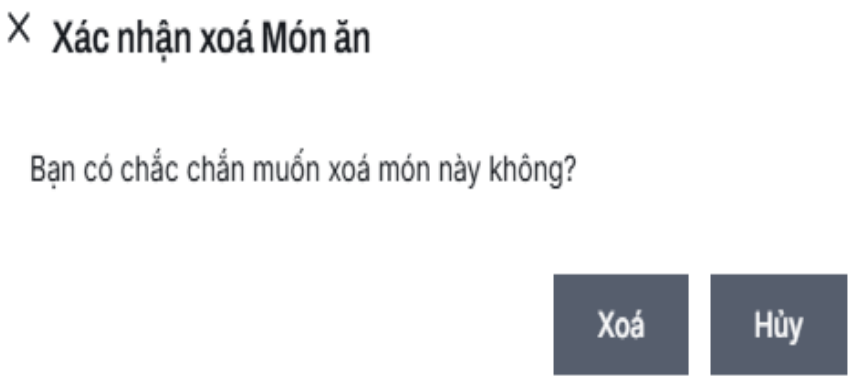


Figure 90: Food delete page of DineGO System

3.9.8 View Orders

Function trigger:

Triggered when the restaurant owner navigates to **Order List** from the left menu.

URL: https://dinego.shop/Order

Function description:

Allows the owner to view and manage all food orders, including order detail, total price, and current status. The owner can filter orders by status and update order progress.

Normal execution case:

Owner views the list of orders → filters by status if needed → updates the order status.

Abnormal execution cases:

- No orders found → display "No orders found" message.
- Update fails → show error message.

Screen layout:

- **Tabs/Filters:** All, Pending Confirmation, Confirmed, Delivering, Completed, Cancelled
- **Table/List:** Columns for Order ID, Customer Name, Order Date, Total Amount, Status, Actions
- **Action:** Dropdown to change status + Update button

Figure 91: View Orders page of DineGO System

3.9.9 View Reservations

Function trigger:

Triggered when the restaurant owner navigates to **Reservation History** from the left menu.

URL: https://dinego.shop/Table/ReservationHistory

Function description:

Allows the owner to view all table reservations, including customer detail, booking date and time, table information, notes, and status. The owner can filter reservations by status.

Normal execution case:

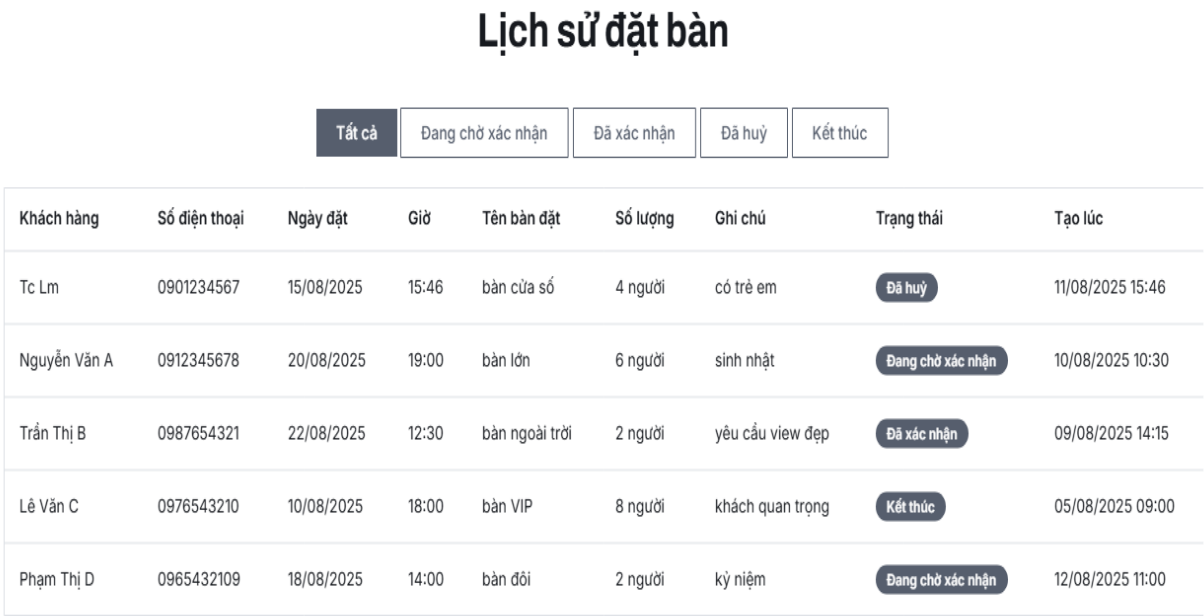
Owner views reservation list → filters by status if needed.

Abnormal execution cases:

- No reservations found → display "No reservations found" message.
- Data load fails → show error message.

Screen layout:

- **Tabs/Filters:** All, Pending Confirmation, Confirmed, Cancelled, Completed
- **Table/List:** Columns for Customer Name, Phone, Booking Date, Booking Time, Table Name, Seats, Notes, Status, Created Time



Khách hàng	Số điện thoại	Ngày đặt	Giờ	Tên bàn đặt	Số lượng	Ghi chú	Trạng thái	Tạo lúc
Tc Lm	0901234567	15/08/2025	15:46	bàn cửa sổ	4 người	có trẻ em	Đã huỷ	11/08/2025 15:46
Nguyễn Văn A	0912345678	20/08/2025	19:00	bàn lớn	6 người	sinh nhật	Đang chờ xác nhận	10/08/2025 10:30
Trần Thị B	0987654321	22/08/2025	12:30	bàn ngoài trời	2 người	yêu cầu view đẹp	Đã xác nhận	09/08/2025 14:15
Lê Văn C	0976543210	10/08/2025	18:00	bàn VIP	8 người	khách quan trọng	Kết thúc	05/08/2025 09:00
Phạm Thị D	0965432109	18/08/2025	14:00	bàn đôi	2 người	kỷ niệm	Đang chờ xác nhận	12/08/2025 11:00

Figure 92: View Reservations page of DineGO System

3.9.10 Revenue by Reservation

Function trigger:

Triggered when the system administrator or restaurant owner navigates to the **Statistics** section.

URL: https://dinego.shop/RestaurantOwner/Chart

Function description:

Displays an overview of the system's business performance, including revenue, orders, customers, and other key metrics. Provides visual charts for revenue trends, order volume, and top performers.

Normal execution case:

Guest views summary statistics and charts for revenue, orders, and ratings.

Abnormal execution cases:

- No data available → display "No data available" message.
- Data load fails → show error message.

Screen layout:

- **Summary Cards:** Total restaurants, Total revenue, Total orders, Total reservations, Total customers, Successful transactions, Failed transactions, Verified licenses, Average rating
- **Charts:** Monthly revenue, Monthly orders, Top 5 restaurants by revenue, Top 5 best-selling foods

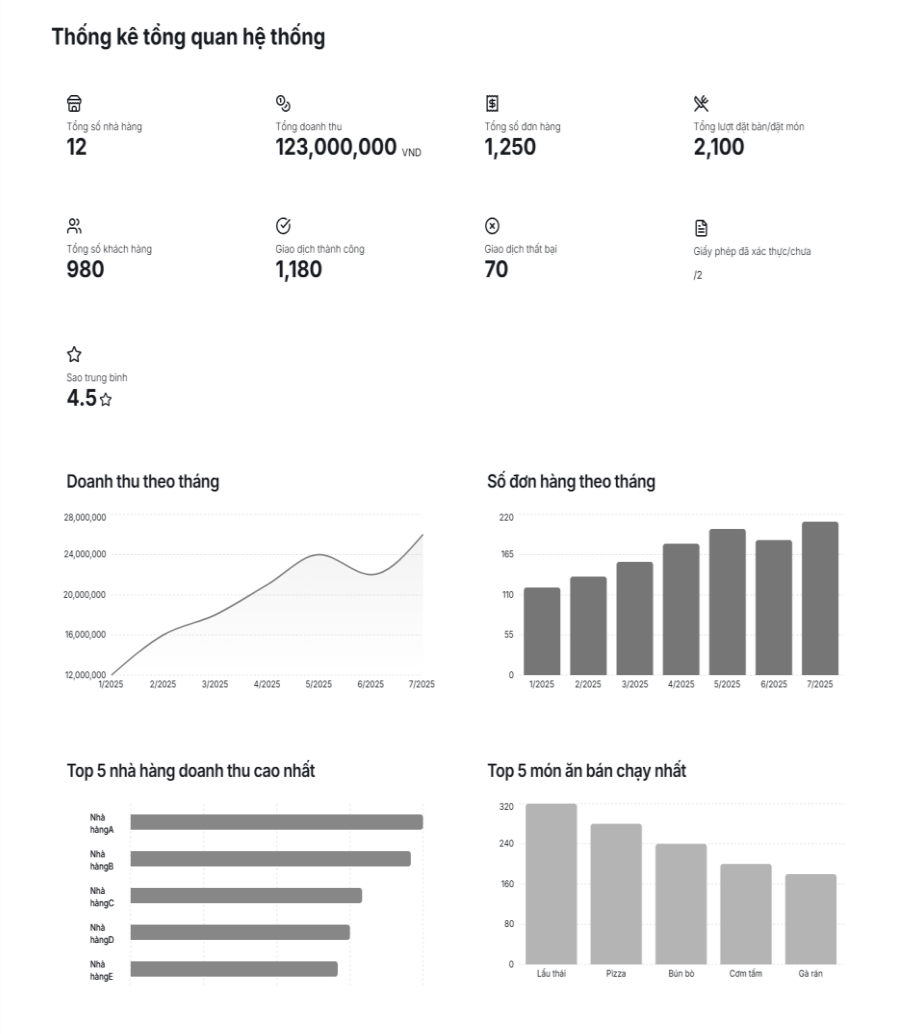


Figure 93: Revenue by Reservation page of DineGO System

3.10 Admin Management

3.10.1 Dashboard

Function trigger:

Triggered when the system administrator accesses the **Dashboard** from the left menu.

URL: https://admin.dinego.shop/

Function description:

Displays key performance indicators (KPIs) and analytical charts for the entire system, including revenue, order volume, customer types, and top-performing restaurants.

Normal execution case:

Admin views KPIs and charts to monitor system performance.

Abnormal execution cases:

- No data available → display “No data available” message.
- Data load fails → show error message.

Screen layout:

- **Charts:**
  - Revenue over time
  - Order volume

- Customer type ratio
- Order status ratio
- Top 5 restaurants by revenue
- Service revenue comparison

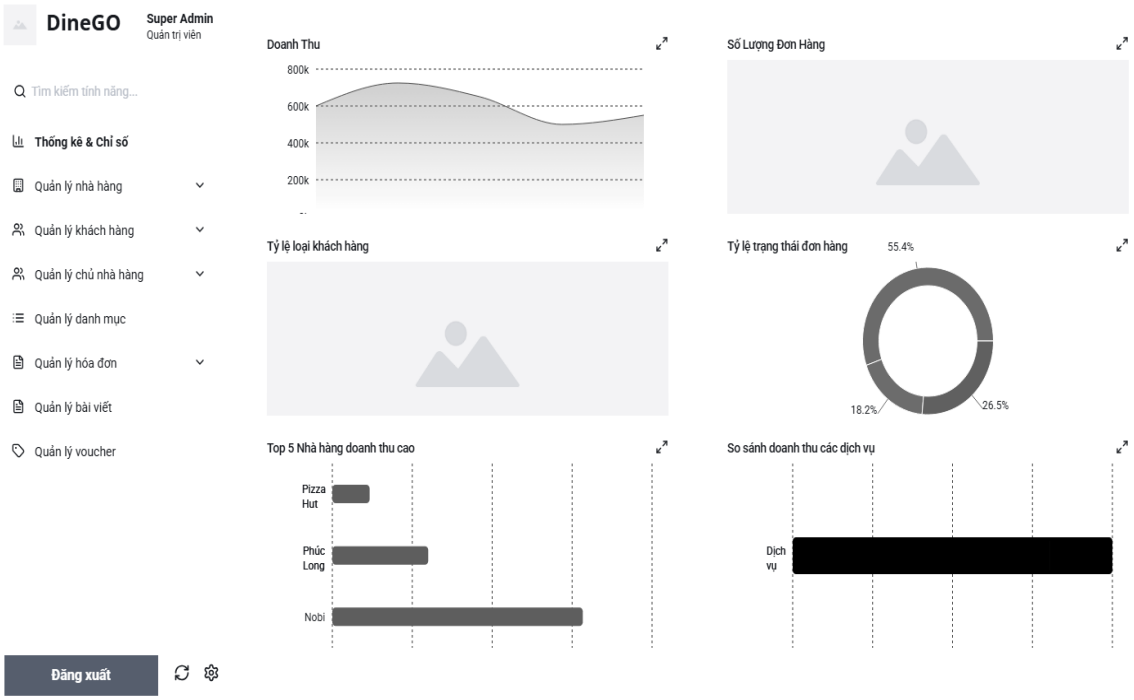


Figure 94: Dashboard page of DineGO System

3.10.2 Manage Restaurants

3.10.2.1 Restaurant List

Function trigger:

Triggered when the system administrator navigates to **Manage Restaurants** from the left menu.

URL: https://admin.dinego.shop/Restaurant

Function description:

Displays a list of all restaurants in the system, including basic information such as name, email, phone number, address, discount, and rating. Admin can add new restaurants or manage existing ones.

Normal execution case:

Admin views restaurant list → can click **Add Restaurant** or access actions for each restaurant.

Abnormal execution cases:

- No restaurants found → display “No restaurants found” message.
- Data load fails → show error message.

Screen layout:

- **Button:** Add Restaurant (top left)

- **Table/List:** Columns for ID, Restaurant Name, Email, Phone Number, Address, Discount, Rating, Actions

Figure 95: Manage Restaurants page of DineGO System

3.10.2.2 Create Restaurant

Function trigger:

Triggered when the system administrator clicks **Add Restaurant** on the Restaurant List page.

URL: https://admin.dinego.shop/Restaurant/AddRestaurant

Function description:

Allows the admin to add a new restaurant by entering detail such as name, category, owner, deposit, contact information, and description, with an option to upload an image.

Normal execution case:

Admin fills in required fields → clicks **Add Restaurant** → system saves the new restaurant and updates the list.

Abnormal execution cases:

- Missing required fields → show validation message.
- Save fails → show error message.

Screen layout:

- **Fields:** Image upload, Restaurant Name, Category, Owner, Deposit, Phone Number, Email, Address, Restaurant Description
- **Buttons:** Add Restaurant, Back

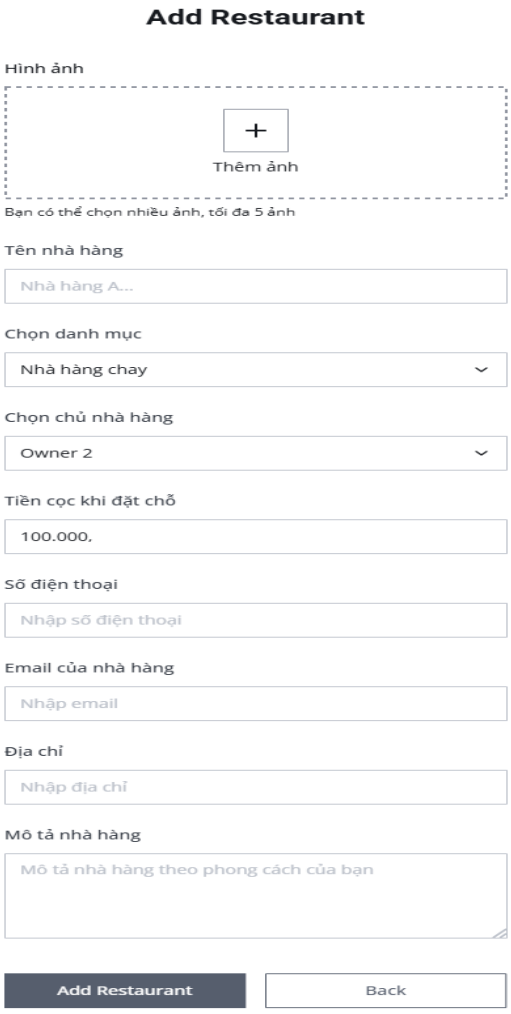


Figure 96: Create restaurants page of DineGO System

3.10.2.3 Edit Restaurant

Function trigger:

Triggered when the system administrator clicks **Edit** on a restaurant in the Restaurant List page.

URL: https://admin.dinego.shop/Restaurant/UpdateRestaurant/3

Function description:

Allows the admin to update restaurant detail such as name, category, owner, deposit, contact information, description, and image.

Normal execution case:

Admin modifies the desired fields → clicks **Update Restaurant** → system saves changes and refreshes the list.

Abnormal execution cases:

- Missing required fields → show validation message.
- Save fails → show error message.

Screen layout:

- **Fields:** Image upload, Restaurant Name, Category, Owner, Deposit, Phone Number, Email, Address, Restaurant Description

- **Buttons:** Update Restaurant, Back

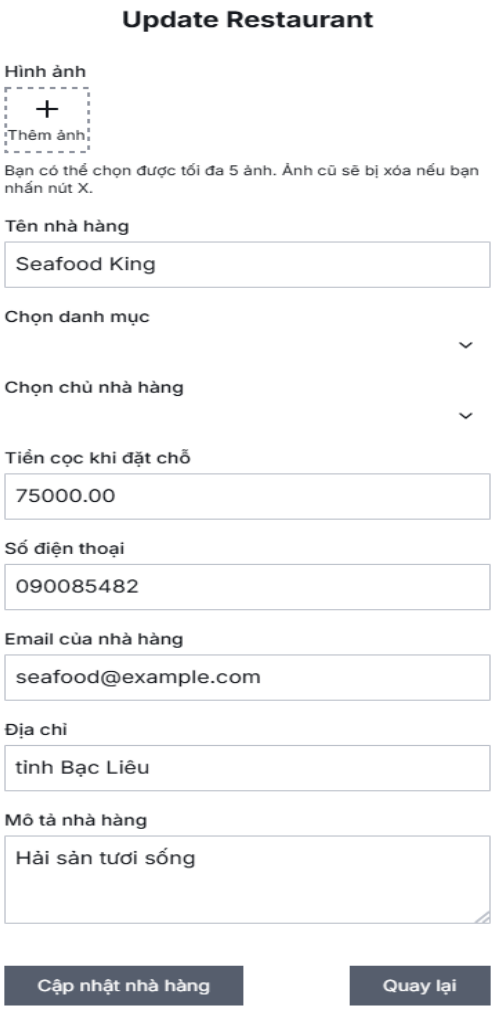


Figure 97: Edit restaurants page of DineGO System

3.10.2.4 Block Restaurant

Function trigger:

Triggered when the system administrator selects **Block** on a restaurant from the Restaurant List page.

Function description:

Allows the admin to block a restaurant by selecting block type, violation type, and entering the reason.

Normal execution case:

Admin fills in required information → clicks **Block** → system blocks the restaurant and updates its status.

Abnormal execution cases:

- Missing required fields → show validation message.
- Block action fails → show error message.

Screen layout:

- **Fields:** Block Type (dropdown), Violation Type (dropdown), Block Reason (text area)
- **Buttons:** Cancel, Block

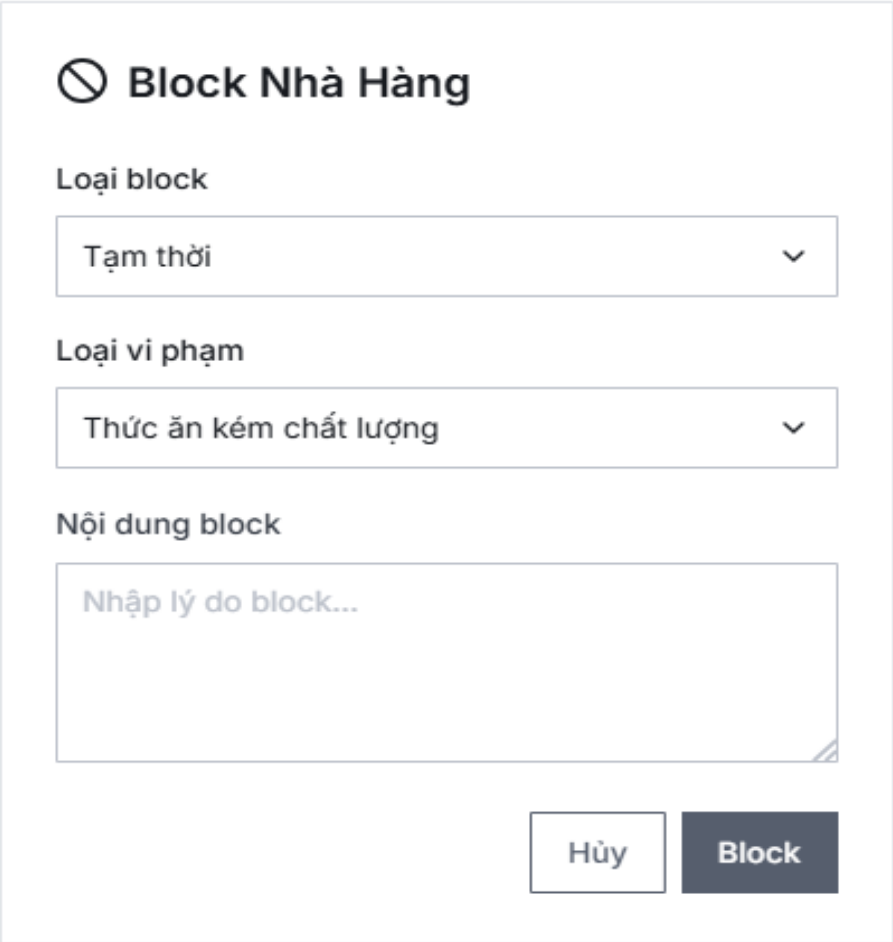


Figure 98: Block restaurants page of DineGO System

3.10.2.5 Delete Restaurant

Function trigger:

Triggered when the system administrator selects **Delete** on a restaurant from the Restaurant List page.

Function description:

Displays a confirmation popup asking the admin to confirm before deleting the selected restaurant.

Normal execution case:

Admin confirms deletion → system removes the restaurant and updates the list.

Abnormal execution cases:

- Delete fails → show error message.
- Restaurant is in use (linked to orders or data) → show warning and prevent deletion.

Screen layout:

- Confirmation popup with message
- **Buttons:** OK, Cancel

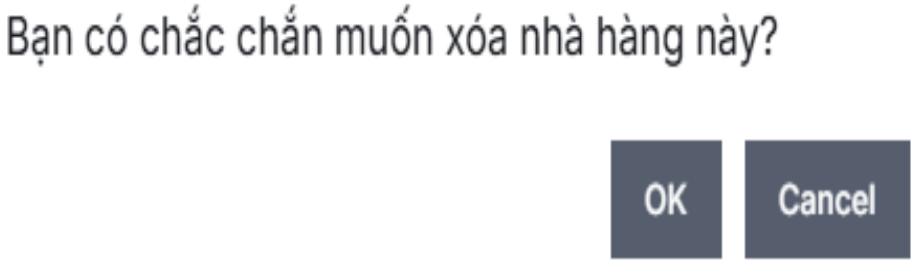


Figure 99: Delete restaurants page of DineGO System

3.10.3 License Approval

Function trigger:

Triggered when the system administrator navigates to **License Approval** from the left menu.

URL: https://admin.dinego.shop/Verification

Function description:

Displays the list of restaurant license verification requests, including restaurant detail, tax code, license number, attached documents, and current approval status. Admin can review and approve or reject requests.

Normal execution case:

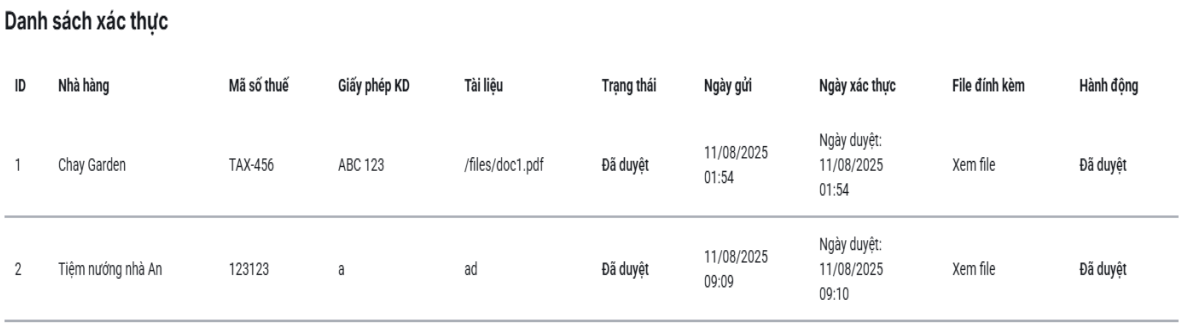
Admin reviews license detail → clicks **Approve** or **Reject** → system updates status.

Abnormal execution cases:

- No requests found → display "No license requests found" message.
- Action fails → show error message.

Screen layout:

- **Table/List:** Columns for ID, Restaurant Name, Tax Code, License Number, Document, Status, Submission Date, Approval Date, Attached File, Actions



Danh sách xác thực

ID	Nhà hàng	Mã số thuế	Giấy phép KD	Tài liệu	Trạng thái	Ngày gửi	Ngày xác thực	File đính kèm	Hành động
1	Chay Garden	TAX-456	ABC 123	/files/doc1.pdf	Đã duyệt	11/08/2025 01:54	Ngày duyệt: 11/08/2025 01:54	Xem file	Đã duyệt
2	Tiệm nướng nhà An	123123	a	ad	Đã duyệt	11/08/2025 09:09	Ngày duyệt: 11/08/2025 09:10	Xem file	Đã duyệt

Figure 100: License Approval page of DineGO System

3.10.4 Manage Customers

3.10.4.1 Customer List

Function trigger:

Triggered when the system administrator navigates to **Manage Customers** from the left menu.

URL: https://admin.dinego.shop/Customer

Function description:

Displays the list of all customers in the system, including profile picture, login name, full name, date of birth, gender, phone number, and email. Admin can add new customers or manage existing ones.

Normal execution case:

Admin views customer list → can click **Add Customer** or access actions for each customer.

Abnormal execution cases:

- No customers found → display "No customers found" message.
- Data load fails → show error message.

Screen layout:

- **Button:** Add Customer (top left)
- **Table/List:** Columns for ID, Avatar, Login Name, Full Name, Date of Birth, Gender, Phone Number, Email, Actions

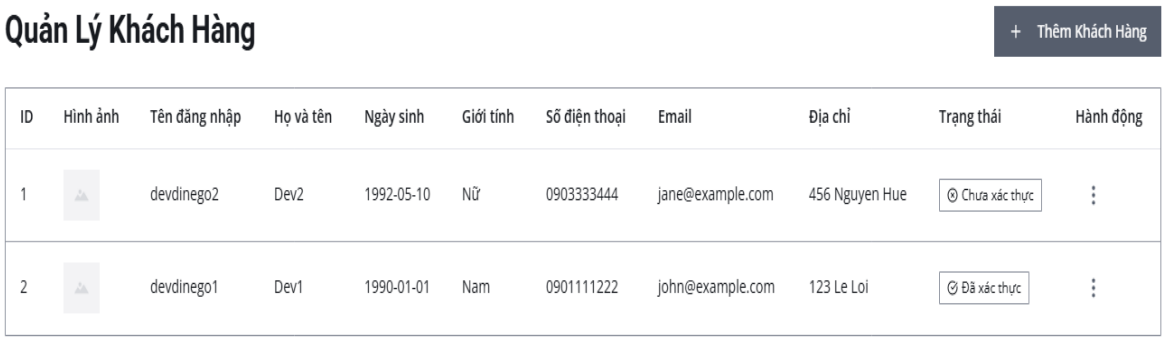


Figure 101: Manage Customers page of DineGO System

3.10.4.2 Create Customer

Function trigger:

Triggered when the system administrator clicks **Add Customer** on the Customer List page.

URL: https://admin.dinego.shop/Customer/AddCustomer

Function description:

Allows the admin to create a new customer account by filling in the required personal and login information, with an option to upload an avatar.

Normal execution case:

Admin completes required fields → clicks **Add Customer** → system saves the new customer and updates the list.

Abnormal execution cases:

- Missing required fields → show validation message.
- Account creation fails → show error message.

Screen layout:

- **Fields:** Avatar upload, Guestname, Password, Confirm Password, Full Name, Email, Phone Number

- **Buttons:** Add Customer, Cancel

Figure 102: Create Customers page of DineGO System

3.10.4.3 Edit Customer

Function trigger:

Triggered when the system administrator clicks **Edit** on a customer in the Customer List page.

URL: https://admin.dinego.shop/Customer/UpdateCustomer/1

Function description:

Allows the admin to update customer detail such as full name, date of birth, gender, phone number, email, address, and avatar.

Normal execution case:

Admin modifies the desired fields → clicks **Update Customer** → system saves changes and refreshes the list.

Abnormal execution cases:

- Missing required fields → show validation message.
- Update fails → show error message.

Screen layout:

- **Fields:** Avatar upload, Full Name, Birthday, Gender, Phone Number, Email, Address
- **Buttons:** Update Customer, Back

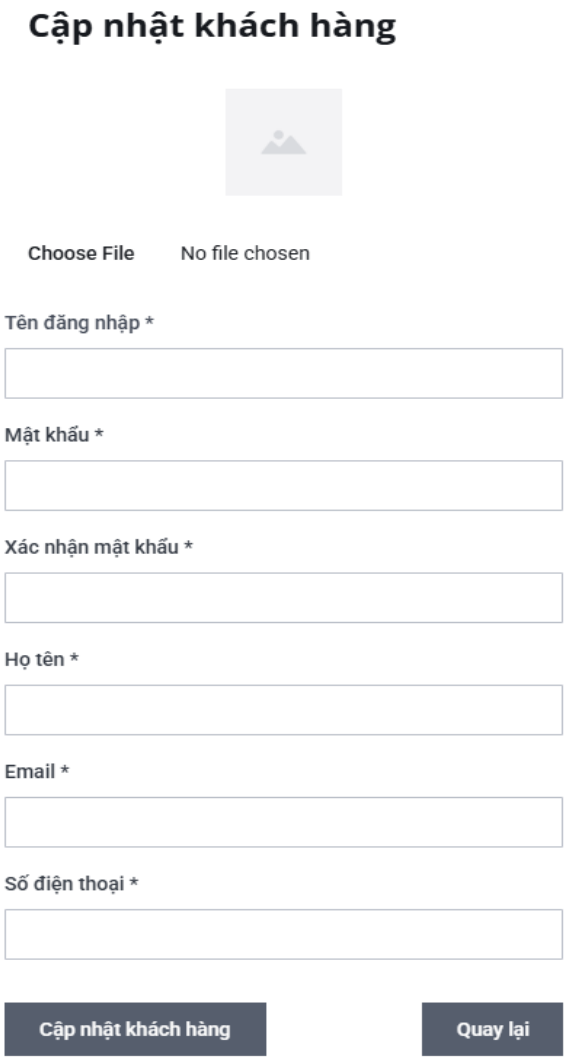


Figure 103: Edit Customers page of DineGO System

3.10.4.4 Block Customer

Function trigger:

Triggered when the system administrator selects the **Block** option for a customer in the Customer List page.

Function description:

Allows the admin to block a customer by specifying the block type, violation type, and block reason.

Normal execution case:

Admin selects block type and violation type → enters reason → clicks **Block** → system updates customer status to blocked.

Abnormal execution cases:

- Missing required selections → show validation message.
- Block action fails → show error message.

Screen layout:

- **Fields:** Block Type (dropdown), Violation Type (dropdown), Block Reason (text area)
- **Buttons:** Cancel, Block

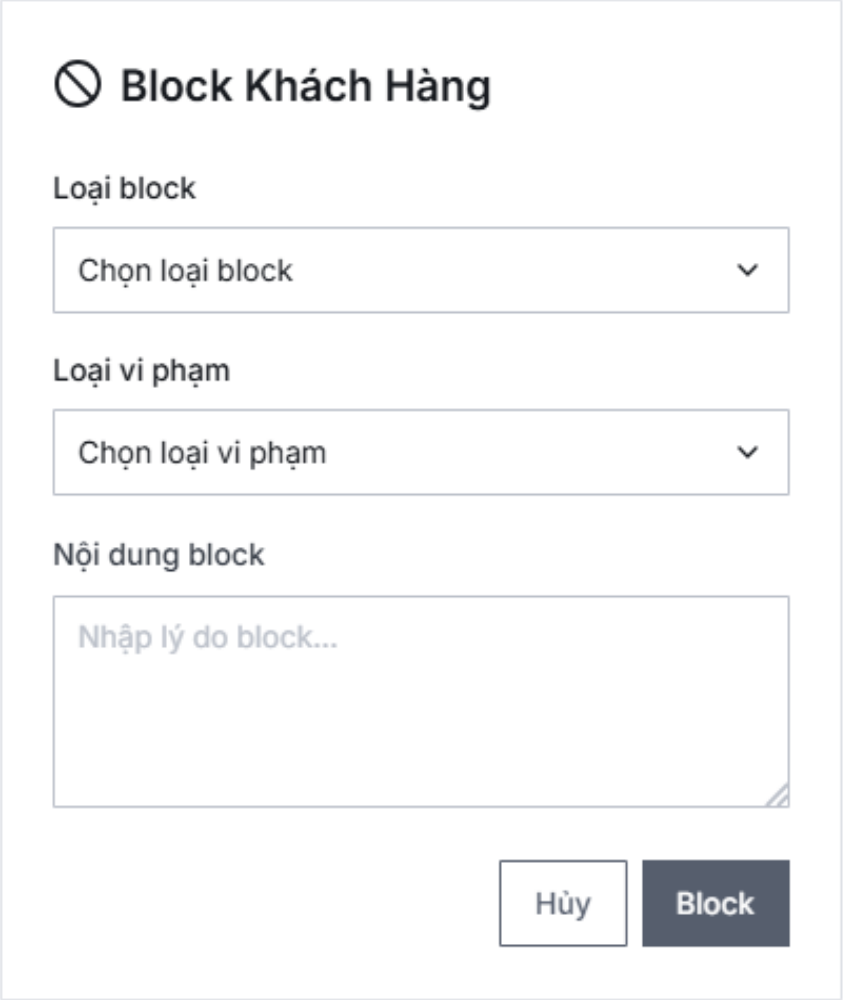


Figure 104: Block Customers page of DineGO System

3.10.4.5 Delete Customer

Function trigger:

Triggered when the system administrator clicks **Delete** for a customer in the Customer List page.

URL: https://admin.dinego.shop/Customer/DeleteCustomer/1

Function description:

Allows the admin to permanently remove a customer's account from the system.

Normal execution case:

Admin reviews customer detail → clicks **Delete Customer** → system confirms deletion and removes the customer.

Abnormal execution cases:

- Deletion fails due to system error → show error message.
- Customer not found → return to customer list.

Screen layout:

- **Fields:** ID, Guestname, Name, Email (read-only)

- **Buttons:** Delete Customer, Back

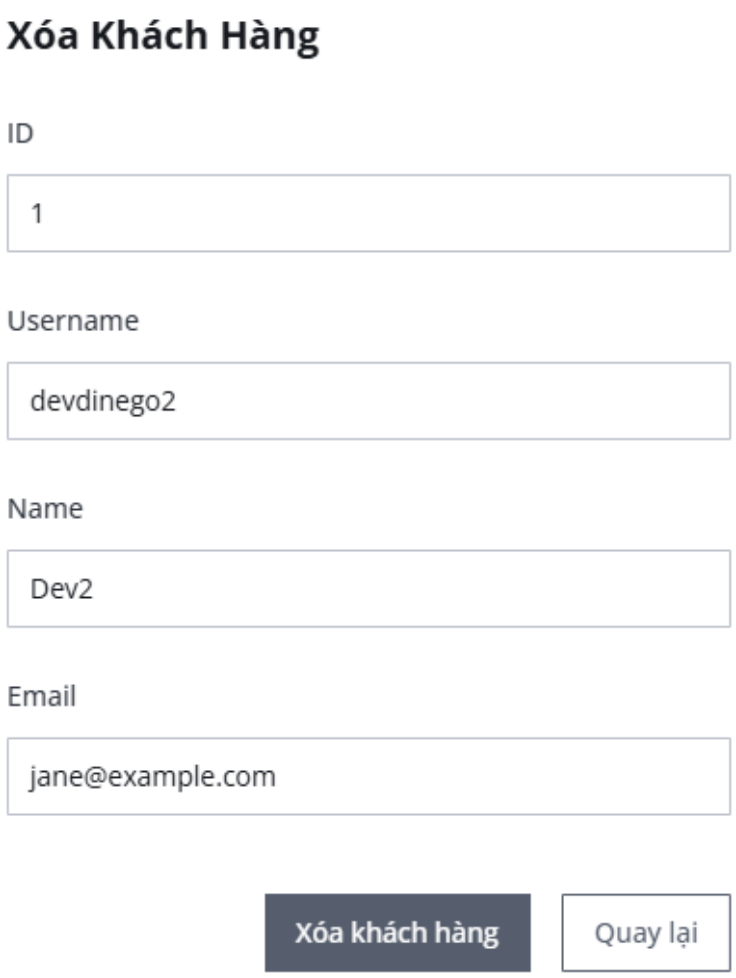


Figure 105: Delete Customers page of DineGO System

3.10.5 Manage Vouchers

3.10.5.1 Voucher List

Function trigger:

Triggered when the system administrator accesses the **Manage Vouchers** page from the admin menu.
URL: https://admin.dinego.shop/Voucher

Function description:

Displays a list of all vouchers in the system, including code, value, type, quantity, start/end dates, status, and applicable targets. Admin can add, edit, or delete vouchers.

Normal execution case:

System loads all vouchers and displays them in a table with relevant detail and action buttons.

Abnormal execution cases:

- No vouchers found → show empty state message.
- Data loading error → show error message.

Screen layout:

- **Columns:** ID, Code, Value, Type, Quantity, Start Date, End Date, Status, Applies To, Actions.

- **Buttons:** Add Voucher, Edit, Delete.

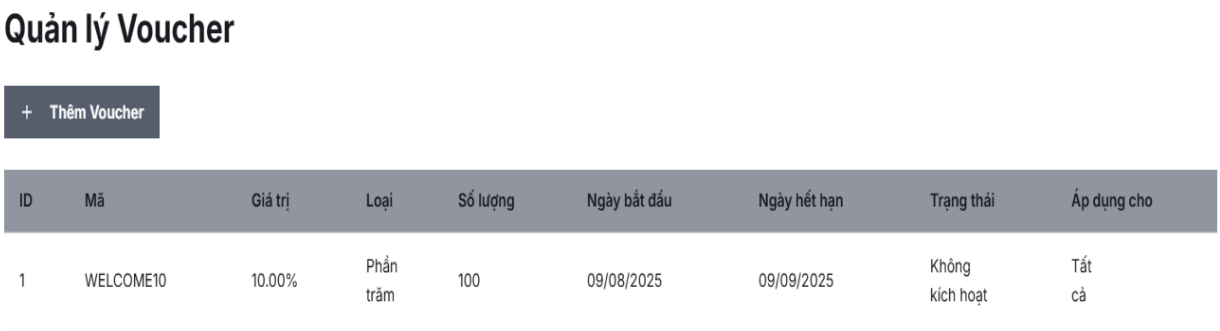


Figure 106: Manage Vouchers page of DineGO System

3.10.5.2 Create Voucher

Function trigger:

Triggered when the administrator clicks the **Add Voucher** button on the Manage Vouchers page.

URL: https://admin.dinego.shop/Voucher/Create

Function description:

Allows the administrator to create a new voucher by specifying code, discount value, discount type, quantity, start/end date, status, and applicable targets.

Normal execution case:

- Admin enters all required detail → clicks **Add** → system saves the voucher → new voucher appears in the list.

Abnormal execution cases:

- Missing required fields → show validation error message.
- Invalid date range (end date before start date) → show error message.
- Duplicate voucher code → show error and prevent saving.

Screen layout:

- **Fields:** Voucher Code, Discount Value, Discount Type, Quantity, Start Date, End Date, Status, Applies To.
- **Buttons:** Add, Cancel.

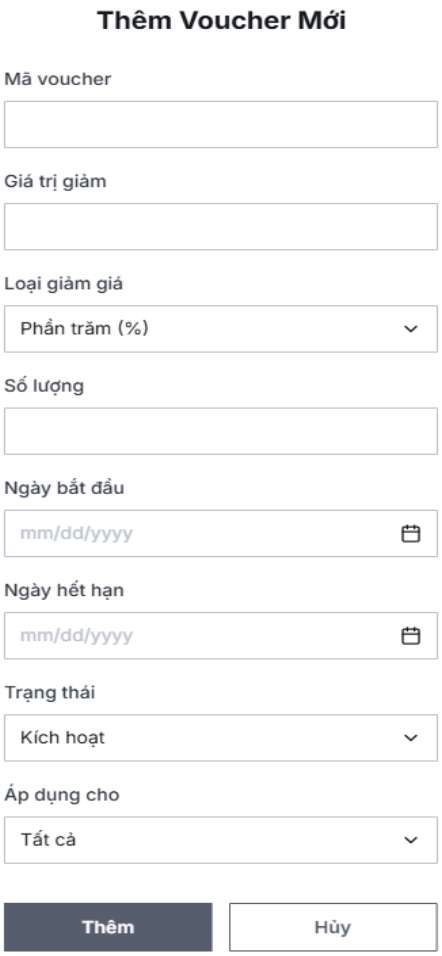


Figure 107: Create Vouchers page of DineGO System

3.10.5.3 Edit Voucher

Function trigger:

Triggered when the administrator clicks the **Edit** button on a voucher in the Manage Vouchers list.

URL: https://admin.dinego.shop/Voucher/Edit/1

Function description:

Allows the administrator to update voucher detail, including code, discount value, discount type, quantity, start/end date, status, and applicable targets.

Normal execution case:

- Admin modifies required fields → clicks **Save** → system updates the voucher → updated detail appear in the list.

Abnormal execution cases:

- Missing required fields → show validation error message.
- Invalid date range (end date before start date) → show error message.
- Duplicate voucher code (if changed) → show error and prevent saving.

Screen layout:

- **Fields:** Voucher Code, Discount Value, Discount Type, Quantity, Start Date, End Date, Status, Applies To.

- **Buttons:** Save, Cancel.

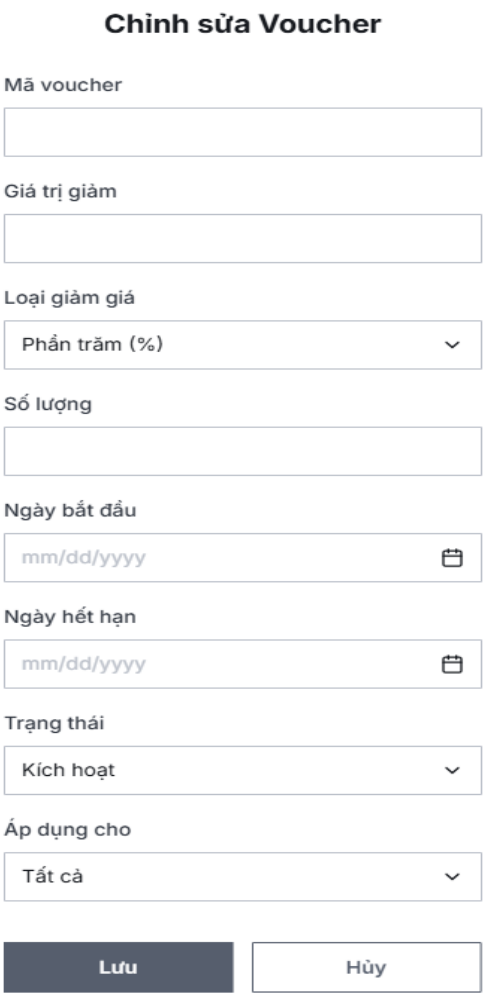


Figure 108: Edit vouchers page of DineGO System

3.10.5.4 Delete Voucher

Function trigger:

Triggered when the administrator clicks the **Delete** button for a voucher in the Manage Vouchers list.

Function description:

Displays a confirmation popup asking the administrator to confirm voucher deletion. If confirmed, the system will permanently remove the voucher from the database.

Normal execution case:

- Admin clicks **Delete** → confirmation popup appears → clicks **OK** → system deletes the voucher → voucher is removed from the list.

Abnormal execution cases:

- Admin clicks **Cancel** → system closes the popup without making any changes.
- System error during deletion → show error message and keep voucher in the list.

Screen layout:

- **Popup message:** "Bạn có chắc chắn muốn xóa voucher này?"
- **Buttons:** OK (confirm deletion), Cancel (close popup without changes).

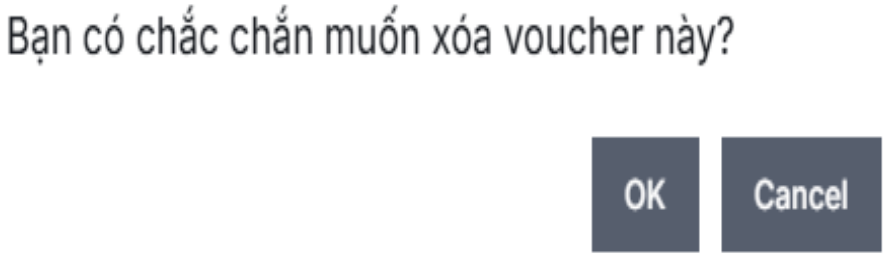


Figure 109: Delete vouchers page of DineGO System

3.10.6 Category Manager

3.10.6.1 Category List

Function trigger:

Triggered when the administrator selects **Manage Categories** from the admin menu.

URL: https://admin.dinego.shop/Category

Function description:

Displays a list of all categories in the system. Each category entry includes its ID, type (Category Name), description, and available actions (Edit/Delete). The administrator can add new categories by clicking the **Add Category** button.

Normal execution case:

- The administrator opens the Category Manager page → the system displays the full list of categories.
- Clicking **Edit** opens the category edit form.
- Clicking **Delete** shows a confirmation popup before deletion.
- Clicking **Add Category** opens the create category form.

Abnormal execution cases:

- Data loading failure → display an error message and an empty list.
- Category deletion failure → display an error message and keep the category in the list.

Screen layout:

- **Table columns:** ID, Category Name, Description, Actions (Edit/Delete).
- **Button:** Add Category.

Figure 110: Category manager page of DineGO System

3.10.6.2 Create Category

Function trigger:

Triggered when the administrator clicks the **Add Category** button on the Category Manager page.

URL:https://admin.dinego.shop/Category/AddCategory

Function description:

Allows the administrator to create a new category by entering the category name and an optional description.

Normal execution case:

1. Admin clicks **Add Category** → the Create Category form is displayed.
2. Admin enters a **Category Name** (required).
3. Admin enters a **Description** (optional).
4. Admin clicks **Add** → the system validates the input, saves the category, and redirects back to the Category Manager page showing the updated list.

Abnormal execution cases:

- Missing Category Name → display a validation message: *"Please enter category name"*.
- Save failure due to server error → display an error message and keep the form open.

Screen layout:

- **Field:** Category Name (required).
- **Field:** Description (optional).
- **Buttons:** Add (submit), Cancel (go back).

Figure 111: Create category page of DineGO System

3.10.6.3 Edit Category

Function trigger:

Triggered when the administrator clicks the **Edit** icon for a category in the Category Manager page.

URL: https://admin.dinego.shop/Category/UpdateCategory/1

Function description:

Allows the administrator to update the category name and description.

Normal execution case:

1. Admin clicks **Edit** → the Edit Category form is displayed, pre-filled with existing category data.
2. Admin updates the **Category Name** and/or **Description**.
3. Admin clicks **Update Category** → the system validates and saves the updated data.
4. The system returns to the Category Manager page with the updated information displayed.

Abnormal execution cases:

- Missing Category Name → display a validation message: *"Please enter category name"*.
- Update failure due to server error → display an error message and keep the form open.

Screen layout:

- **Field:** Category ID (read-only).
- **Field:** Category Name (required).
- **Field:** Description (optional).
- **Buttons:** Update Category (submit changes), Cancel (return to Category Manager).

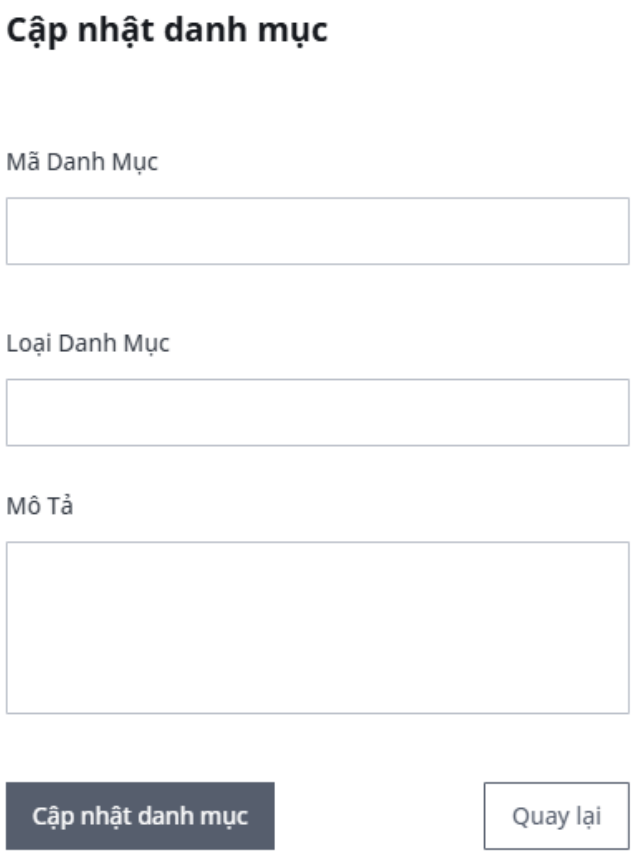


Figure 112: Edit category page of DineGO System

3.10.6.4 Delete Category

Function trigger:

Triggered when the admin clicks the **Delete** icon in the Category Management page.

URL:https://admin.dinego.shop/Category/DeleteCategory/1

Function description:

Allows the admin to delete a selected category from the system after confirmation.

Normal execution case:

Admin reviews category detail → clicks **Delete Category** → confirms action → system removes the category and updates the list.

Abnormal execution cases:

- Category not found → show error message
- Deletion fails due to related data constraints → show error and cancel action.

Screen layout:

- Field: Category ID (read-only)
- Field: Category Type (read-only)
- Field: Description (read-only)
- Button: **Delete Category** (confirm deletion)
- Button: **Back** (cancel and return)

Figure 113: Delete category page of DineGO System

3.10.7 Manage Notifications

3.10.7.1 Notification List

Function trigger:

Triggered when the administrator selects **Manage Notifications** from the sidebar menu.

URL:https://admin.dinego.shop/Notification

Function description:

Displays a list of all notifications in the system, including their title, content, type, creation date, and action buttons for editing or deleting.

Normal execution case:

System loads the notification list → admin can view all notifications → admin may click **Edit** or **Delete** for a selected notification.

Abnormal execution cases:

- No notifications available → show an empty list message
- Data load fails → show an error message

Screen layout:

- **Columns:** ID, Title, Content, Type, Creation Date, Actions.
- **Buttons:** + Add Notification, Edit, Delete.

Figure 114: Manage Notifications page of DineGO System

3.10.7.2 Create Notification

Function trigger:

Triggered when the administrator clicks the **+ Add Notification** button in the Manage Notifications list.

URL:https://admin.dinego.shop/Notification/Create

Function description:

Allows the administrator to create a new notification by entering its title, content, and type.

Normal execution case:

Admin enters required fields → clicks **Add** → system validates input → saves the notification → new notification appears in the list.

Abnormal execution cases:

- Missing required fields → show validation error message.
- Data save fails → show error message.

Screen layout:

- **Fields:** Title, Content, Type.
- **Buttons:** Add, Cancel.

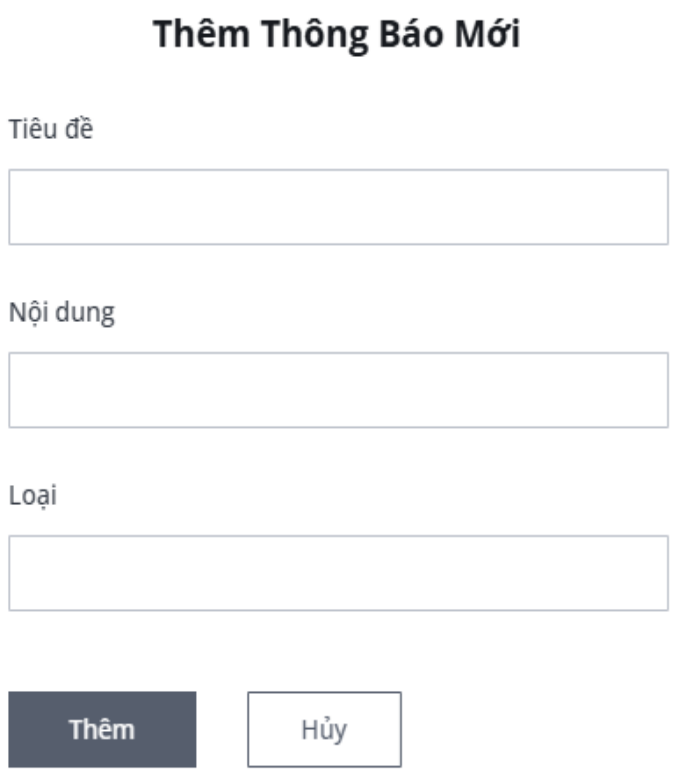


Figure 115: Create Notifications page of DineGO System

3.10.7.3 Edit Notification

Function trigger:

Triggered when the administrator clicks the **Edit** button for a notification in the Manage Notifications list.

URL:https://admin.dinego.shop/Notification/Edit/2

Function description:

Allows the administrator to update an existing notification's title, content, and type.

Normal execution case:

Admin modifies one or more fields → clicks **Save** → system validates input → updates the notification → updated notification appears in the list.

Abnormal execution cases:

- Missing required fields → show validation error message.
- Data save fails → show error message.

Screen layout:

- **Fields:** Title, Content, Type.
- **Buttons:** Save, Cancel.

Figure 116: Edit notifications page of DineGO System

3.10.7.4 Delete Notification

Function trigger:

Triggered when the administrator clicks the **Delete** button for a notification in the Manage Notifications list.

Function description:

Displays a confirmation popup asking the administrator to confirm deletion of the selected notification.

Normal execution case:

Admin clicks **Delete** → confirmation popup appears → admin clicks **OK** → system deletes the notification → updated list is displayed without the deleted item.

Abnormal execution cases:

- Notification not found → show error message.
- Deletion fails due to system error → show error message.

Screen layout:

- **Popup message:** "Are you sure you want to delete this notification?"
- **Buttons:** OK, Cancel.

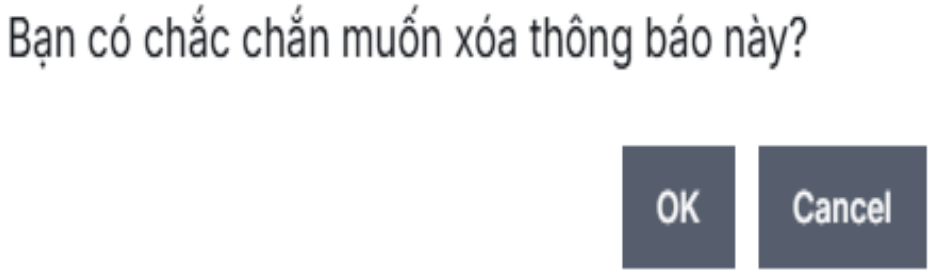


Figure 117: Delete notifications page of DineGO System

3.10.8 Restaurant owner manager

3.10.8.1 Restaurant Owner List

Function trigger:

Triggered when the administrator selects **Manage Restaurant Owners** from the sidebar menu.

URL:https://admin.dinego.shop/RestaurantOwner

Function description:

Displays a list of all restaurant owners, including associated customer account, restaurant name, creation date, and action buttons for management.

Normal execution case:

System loads the restaurant owner list → admin can view detail → admin may click an action button (e.g., Edit, Delete, View Detail).

Abnormal execution cases:

- No restaurant owners available → show an empty list message.
- Data load fails → show error message.

Screen layout:

- **Columns:** ID, Customer, Restaurant Name, Creation Date, Actions.
- **Buttons:** + Add Restaurant Owner, Edit, Delete.

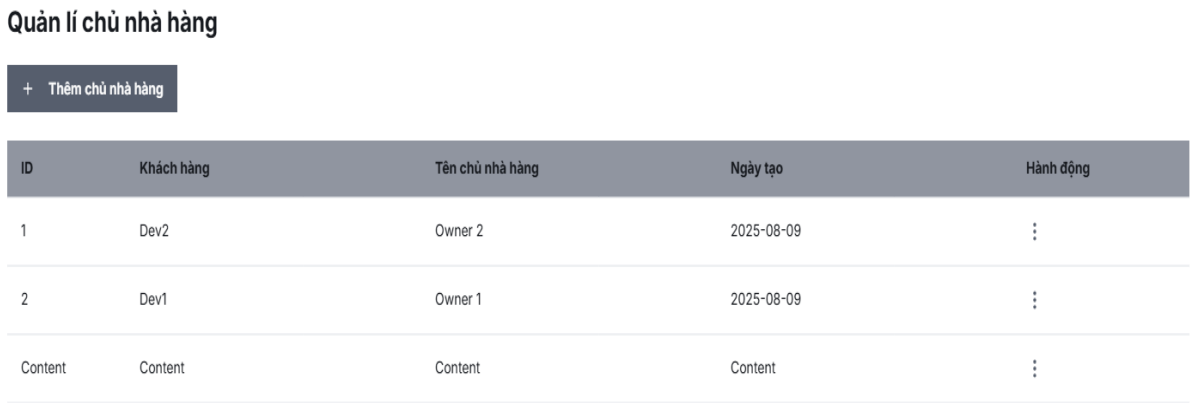


Figure 118: Restaurant owner page of DineGO System

3.10.8.2 Create Restaurant Owner

Function trigger:

Triggered when the administrator clicks the **+ Add Restaurant Owner** button in the Restaurant Owner List page.

URL:https://admin.dinego.shop/RestaurantOwner/Create

Function description:

Allows the administrator to create a new restaurant owner by selecting an existing customer and entering the restaurant owner's name.

Normal execution case:

Admin selects a customer from the dropdown → enters the restaurant owner name → clicks **Add** → system saves the new owner → new owner appears in the list.

Abnormal execution cases:

- Missing required fields → show validation error message.
- Selected customer already assigned as a restaurant owner → show error message.
- Data save fails → show error message.

Screen layout:

- **Fields:** Select Customer (dropdown), Restaurant Owner Name.
- **Buttons:** Add, Cancel.

Figure 119: Create restaurant owner page of DineGO System

3.10.8.3 Edit Restaurant Owner

Function trigger:

Triggered when the administrator clicks the **Edit** button for a restaurant owner in the Restaurant Owner List page.

URL:https://admin.dinego.shop/RestaurantOwner/Edit/1

Function description:

Allows the administrator to update the detail of a restaurant owner, including the associated customer and the restaurant owner's name.

Normal execution case:

Admin changes one or more fields → clicks **Save** → system validates input → updates the restaurant owner → updated detail appear in the list.

Abnormal execution cases:

- Missing required fields → show validation error message.
- Selected customer already assigned as a restaurant owner (if changed) → show error message.
- Data save fails → show error message.

Screen layout:

- **Fields:** Select Customer (dropdown), Restaurant Owner Name.
- **Buttons:** Save, Cancel.

Figure 120: Edit restaurant owner page of DineGO System

3.10.8.4 Delete Restaurant Owner

Function trigger:

Triggered when the administrator clicks the **Delete** button for a restaurant owner in the Restaurant Owner List page.

Function description:

Displays a confirmation popup asking the administrator to confirm deletion of the selected restaurant owner.

Normal execution case:

Admin clicks **Delete** → confirmation popup appears → admin clicks **OK** → system deletes the restaurant owner → updated list is displayed without the deleted entry.

Abnormal execution cases:

- Restaurant owner not found → show error message.
- Deletion fails due to system error → show error message.

Screen layout:

- **Popup message:** "Are you sure you want to delete this restaurant owner?"
- **Buttons:** OK, Cancel.

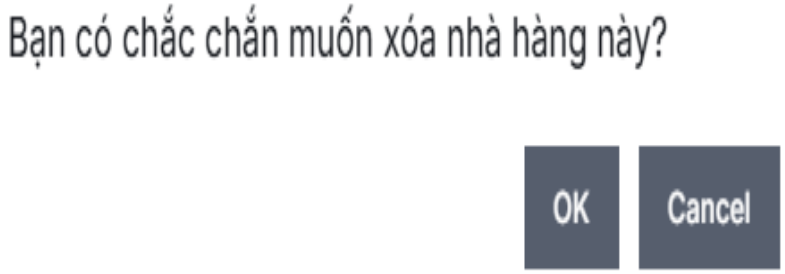


Figure 121: Delete restaurant owner page of DineGO System

3.10.9 Order manager

3.10.9.1 Food Order List

Function trigger:

Triggered when the administrator selects **Manage Orders** from the sidebar menu.

URL:https://admin.dinego.shop/Order

Function description:

Displays a list of all food orders in the system, including order ID, customer, restaurant, order date, status, total amount, discount, and action buttons for order management.

Normal execution case:

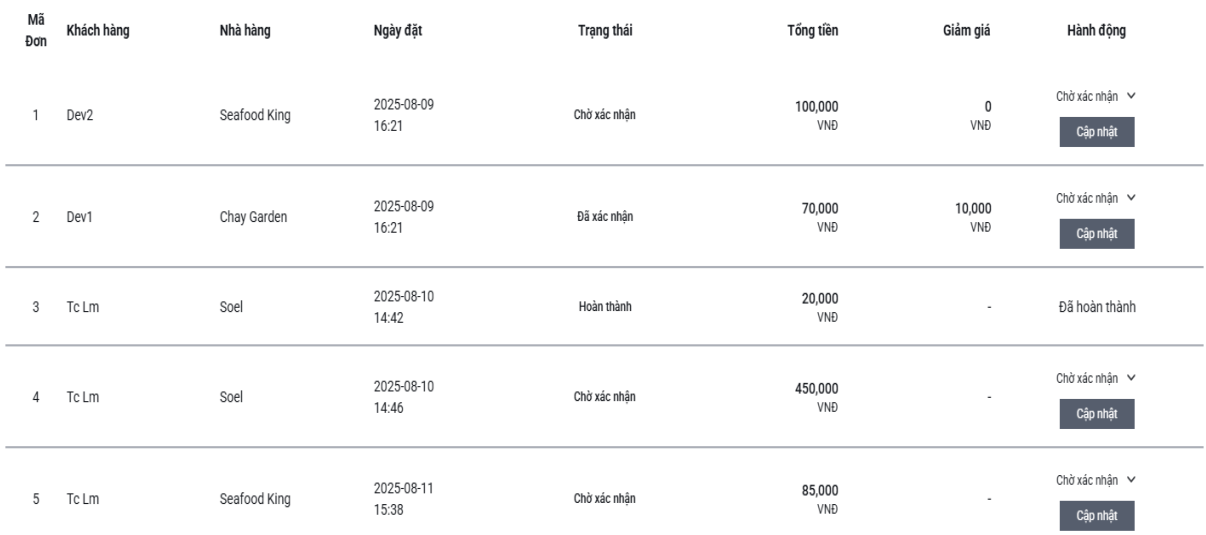
System loads the food order list → admin can view all orders → admin may click an action button (e.g., View Detail, Update Status, Cancel Order).

Abnormal execution cases:

- No orders available → show an empty list message.
- Data load fails → show error message.

Screen layout:

- **Columns:** Order ID, Customer, Restaurant, Order Date, Status, Total Amount, Discount, Actions.
- **Buttons:** View Detail, Update Status, Cancel Order.



Mã Đơn	Khách hàng	Nhà hàng	Ngày đặt	Trạng thái	Tổng tiền	Giảm giá	Hành động
1	Dev2	Seafood King	2025-08-09 16:21	Chờ xác nhận	100,000 VNĐ	0 VNĐ	Chờ xác nhận Cập nhật
2	Dev1	Chay Garden	2025-08-09 16:21	Đã xác nhận	70,000 VNĐ	10,000 VNĐ	Chờ xác nhận Cập nhật
3	Tc Lm	Soel	2025-08-10 14:42	Hoàn thành	20,000 VNĐ	-	Đã hoàn thành
4	Tc Lm	Soel	2025-08-10 14:46	Chờ xác nhận	450,000 VNĐ	-	Chờ xác nhận Cập nhật
5	Tc Lm	Seafood King	2025-08-11 15:38	Chờ xác nhận	85,000 VNĐ	-	Chờ xác nhận Cập nhật

Figure 122: Food order page of DineGO System

3.10.9.2 table Order List

Function trigger:

Triggered when the administrator selects **Manage Table Orders** from the sidebar menu.

URL:https://admin.dinego.shop/Reservation

Function description:

Displays a list of all table reservations in the system, including reservation ID, restaurant, customer, status, date & time, and additional notes.

Normal execution case:

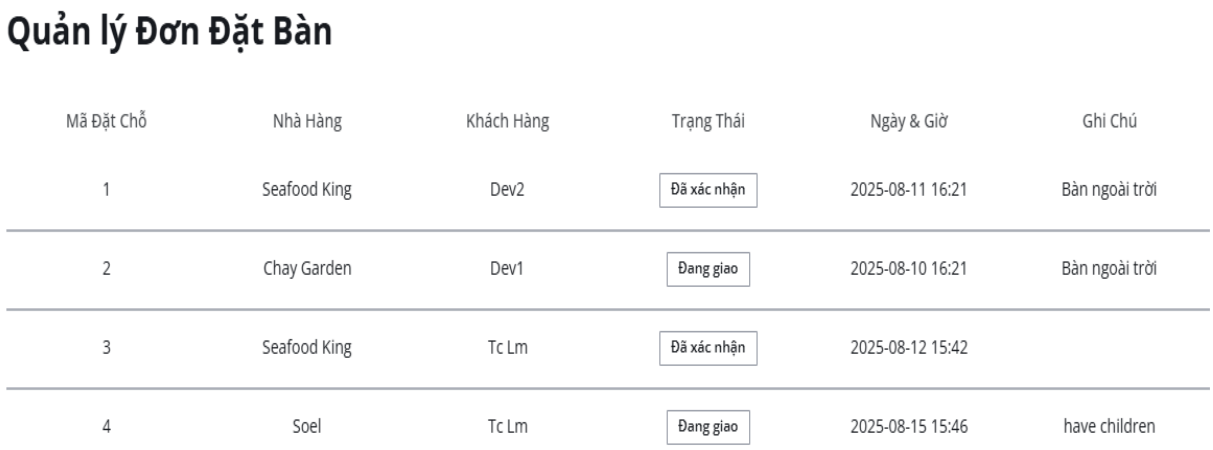
System loads the table order list → admin can view reservation detail → admin may update reservation status or view more detail.

Abnormal execution cases:

- No reservations available → show an empty list message.
- Data load fails → show error message.

Screen layout:

- **Columns:** Reservation ID, Restaurant, Customer, Status, Date & Time, Notes.
- **Buttons:** View Detail, Update Status.



Quản lý Đơn Đặt Bàn

Mã Đặt Chỗ	Nhà Hàng	Khách Hàng	Trạng Thái	Ngày & Giờ	Ghi Chú
1	Seafood King	Dev2	Đã xác nhận	2025-08-11 16:21	Bàn ngoài trời
2	Chay Garden	Dev1	Đang giao	2025-08-10 16:21	Bàn ngoài trời
3	Seafood King	Tc Lm	Đã xác nhận	2025-08-12 15:42	
4	Soel	Tc Lm	Đang giao	2025-08-15 15:46	have children

Figure 123: Table order page of DineGO System

3.10.9.3 Payment Order List

Function trigger:

Triggered when the administrator selects **Payment Orders** from the sidebar menu.

URL: https://admin.dinego.shop/Payment

Function description:

Displays a list of all payment transactions, including customer, amount, creation date, status, and action buttons for invoice printing.

Normal execution case:

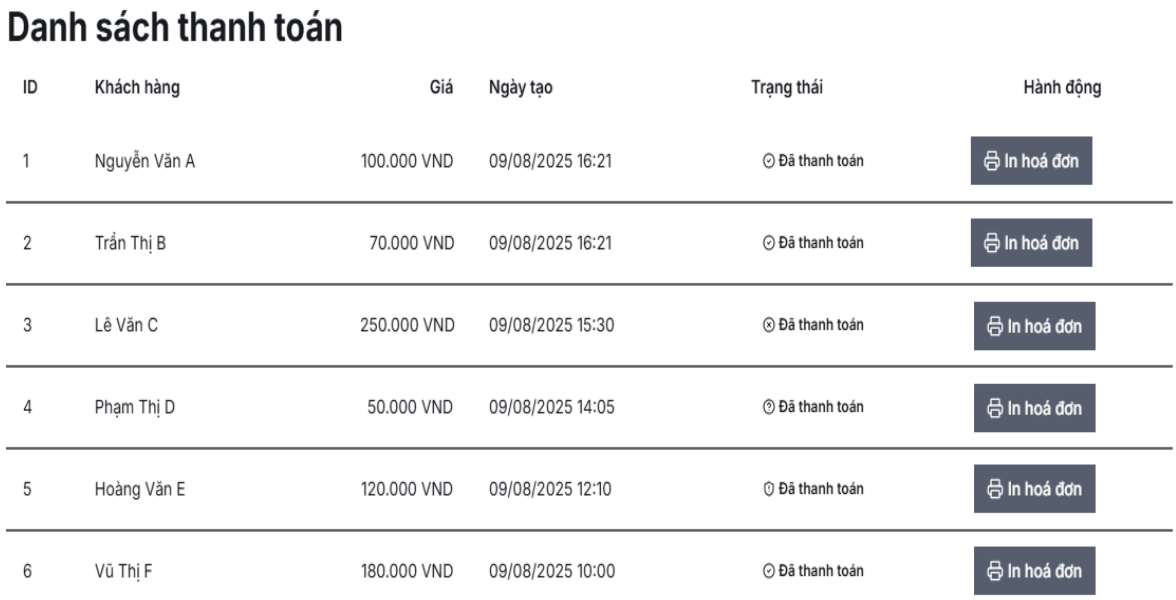
System loads the payment order list → admin can view all payments → admin may click **Print Invoice** for completed payments.

Abnormal execution cases:

- No payment orders available → show an empty list message.
- Data load fails → show error message.

Screen layout:

- **Columns:** ID, Customer, Amount, Creation Date, Status, Actions.
- **Buttons:** Print Invoice.



Danh sách thanh toán

ID	Khách hàng	Giá	Ngày tạo	Trạng thái	Hành động
1	Nguyễn Văn A	100.000 VND	09/08/2025 16:21	Đã thanh toán	In hoá đơn
2	Trần Thị B	70.000 VND	09/08/2025 16:21	Đã thanh toán	In hoá đơn
3	Lê Văn C	250.000 VND	09/08/2025 15:30	Đã thanh toán	In hoá đơn
4	Phạm Thị D	50.000 VND	09/08/2025 14:05	Đã thanh toán	In hoá đơn
5	Hoàng Văn E	120.000 VND	09/08/2025 12:10	Đã thanh toán	In hoá đơn
6	Vũ Thị F	180.000 VND	09/08/2025 10:00	Đã thanh toán	In hoá đơn

Figure 124: Payment order page of DineGO System

3.10.9.4 Payment Order Detail

Function trigger:

Triggered when the administrator clicks the **Print Invoice** or **View Detail** button for a payment order in the Payment Order List page.

URL: https://admin.dinego.shop/Payment/PrintInvoice/1

Function description:

Displays detailed information for a payment order, including invoice detail, customer information, status, total payment amount, and an option to print the invoice.

Normal execution case:

Admin opens payment order detail → system displays all related information → admin may click **Print Invoice** to generate a printable receipt.

Abnormal execution cases:

- Payment order not found → show error message.
- Data retrieval fails → show error message.

Screen layout:

- **Sections:**
  - Invoice Information: Invoice ID, Creation Date, Status.
  - Customer Information: Customer ID, Payment Date.
  - Total Payment Amount.
- **Buttons:** Print Invoice.

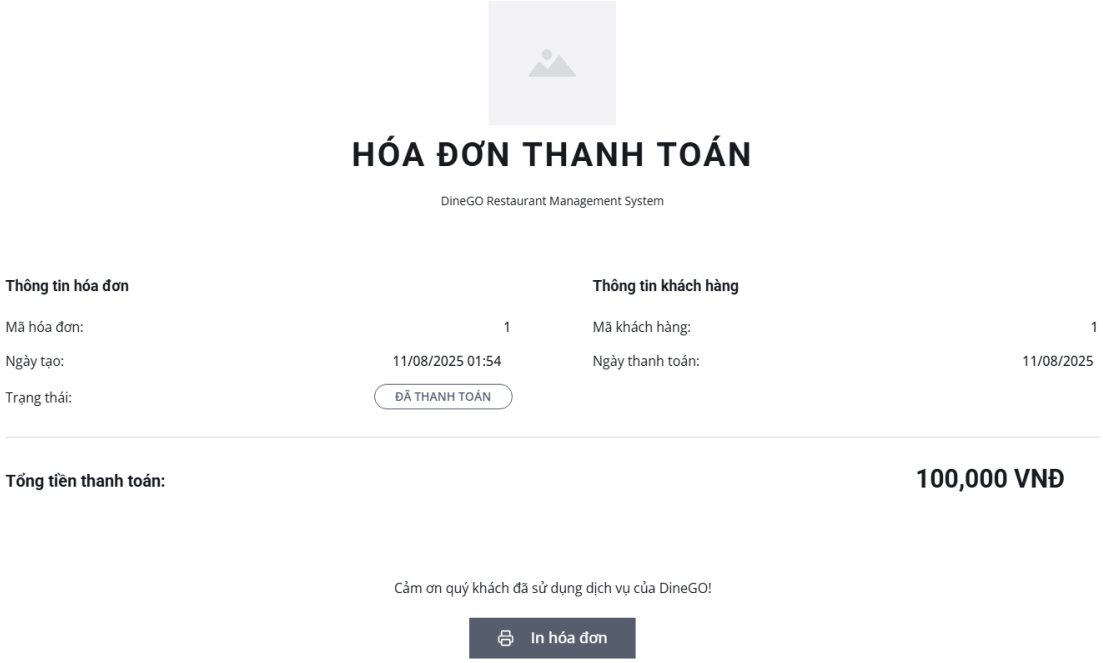


Figure 125: Payment detail page of DineGO System

3.10.10 Blog Manager

3.10.10.1 Blog List

Function trigger:

Triggered when the administrator selects **Manage Blog Posts** from the sidebar menu.

URL:https://admin.dinego.shop/Blog

Function description:

Displays a list of all blog posts in the system, including image, ID, title, date, content, and action buttons for editing or deleting.

Normal execution case:

System loads the blog list → admin can view all blog posts → admin may click **Edit** or **Delete** for a specific post.

Abnormal execution cases:

- No blog posts available → show an empty list message.
- Data load fails → show error message.

Screen layout:

- **Columns:** Image, ID, Title, Date, Content, Actions.
- **Buttons:** + Add Blog Post, Edit, Delete.

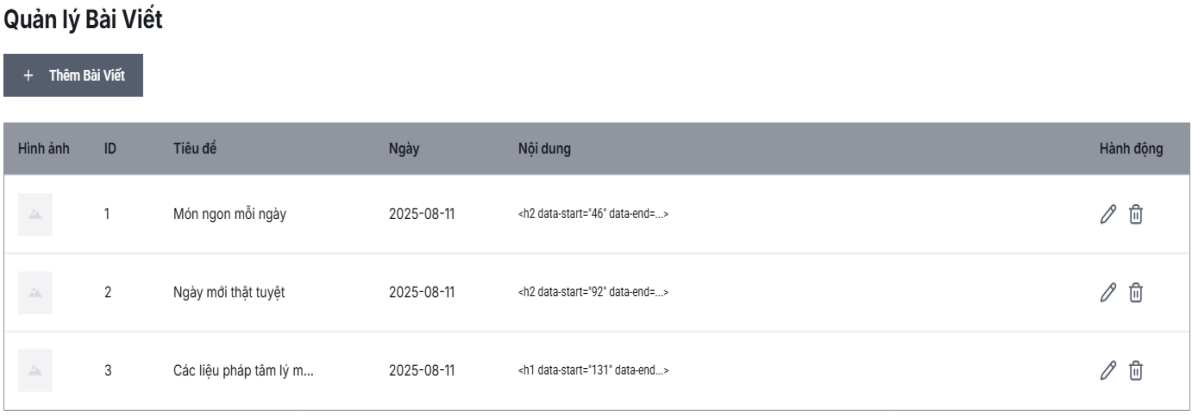


Figure 126: Blog manager page of DineGO System

3.10.10.2 Create Blog

Function trigger:

Triggered when the administrator clicks the **+ Add Blog Post** button in the Blog List page.

URL: https://admin.dinego.shop/Blog/AddBlog

Function description:

Allows the administrator to create a new blog post by entering a title, selecting a thumbnail image, and adding blog content.

Normal execution case:

Admin enters the blog title → uploads a thumbnail image → enters the content → clicks **Add Blog** → system saves the post → new blog appears in the list.

Abnormal execution cases:

- Missing required fields → show validation error message.
- Image upload fails → show error message.
- Data save fails → show error message.

Screen layout:

- **Fields:** Title, Thumbnail Image (file upload), Content (text editor).
- **Buttons:** Add Blog, Cancel.

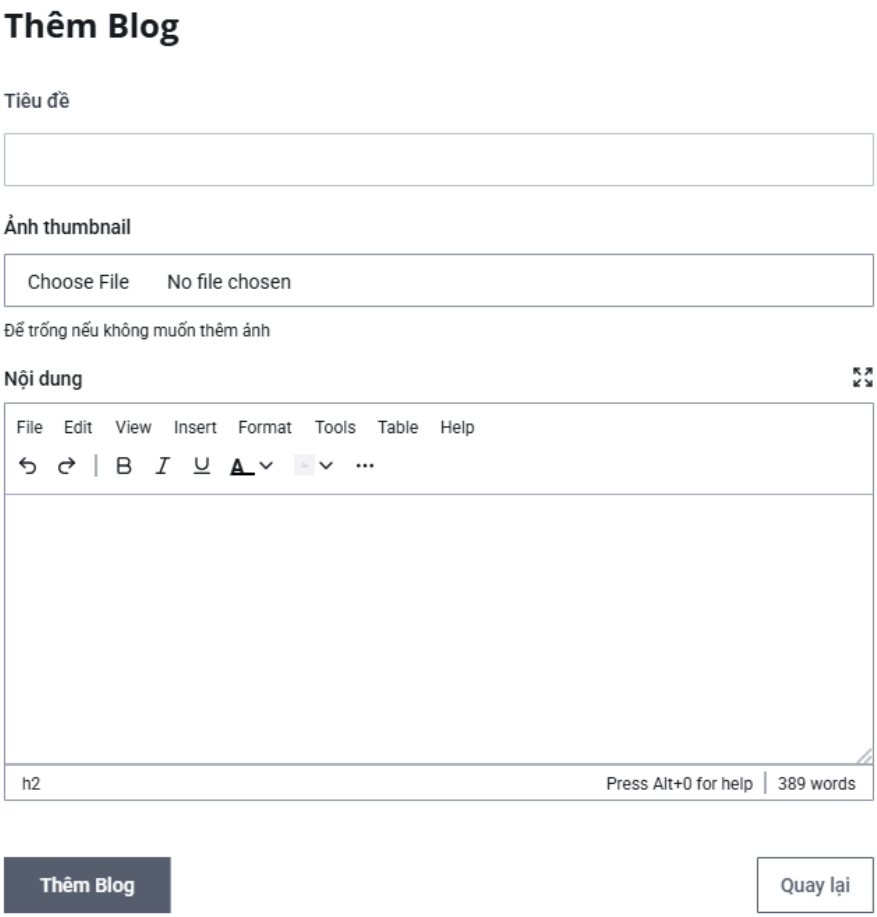


Figure 127: Create blog page of DineGO System

3.10.10.3 Edit Blog

Function trigger:

Triggered when the administrator clicks the **Edit** button for a blog post in the Blog List page.

URL: https://admin.dinego.shop/Blog/UpdateBlog/1

Function description:

Allows the administrator to update the detail of an existing blog post, including the title, thumbnail image, and content.

Normal execution case:

Admin changes one or more fields → clicks **Update Blog** → system validates input → updates the blog post → updated detail appear in the list.

Abnormal execution cases:

- Missing required fields → show validation error message.
- Image upload fails → show error message.
- Data save fails → show error message.

Screen layout:

- **Fields:** Title, Thumbnail Image (file upload), Content (text editor).
- **Buttons:** Update Blog, Cancel.

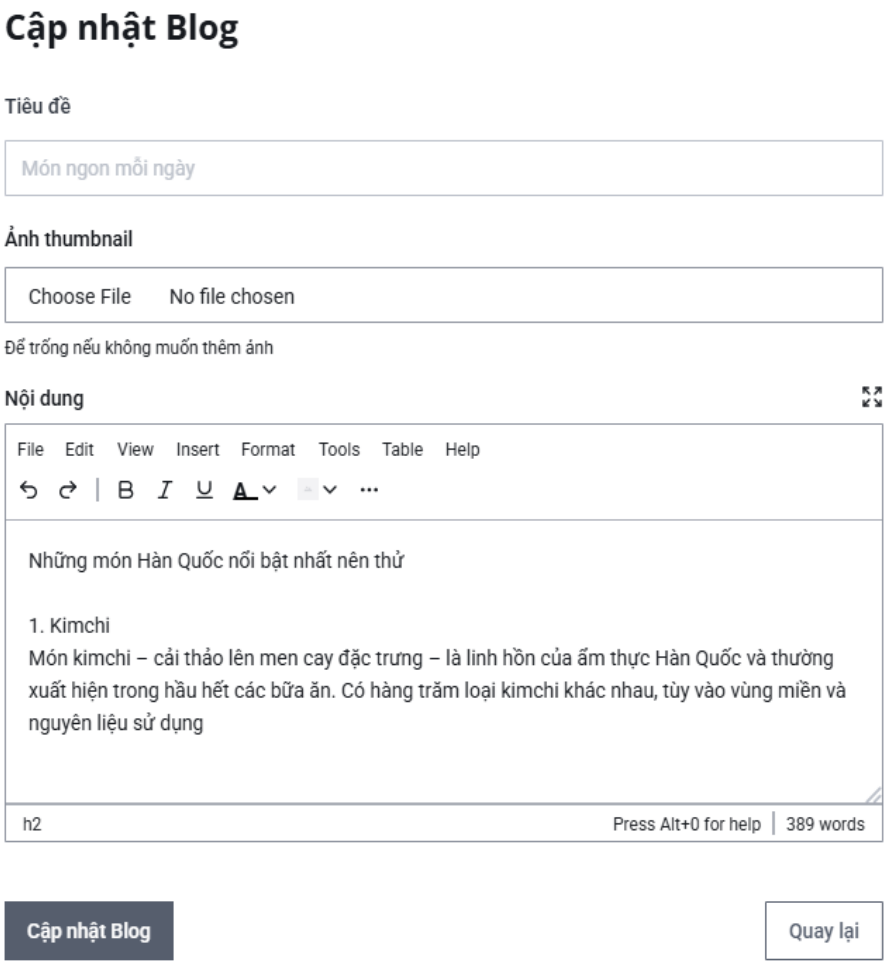


Figure 128: Edit blog page of DineGO System

3.10.10.4 Delete Blog

Function trigger:

Triggered when the administrator clicks the **Delete** button for a blog post in the Blog List page.

Function description:

Displays a confirmation popup asking the administrator to confirm deletion of the selected blog post.

Normal execution case:

Admin clicks **Delete** → confirmation popup appears → admin clicks **OK** → system deletes the blog post → updated list is displayed without the deleted entry.

Abnormal execution cases:

- Blog post not found → show error message.
- Deletion fails due to system error → show error message.

Screen layout:

- **Popup message:** "Are you sure you want to delete this blog post?"
- **Buttons:** OK, Cancel.

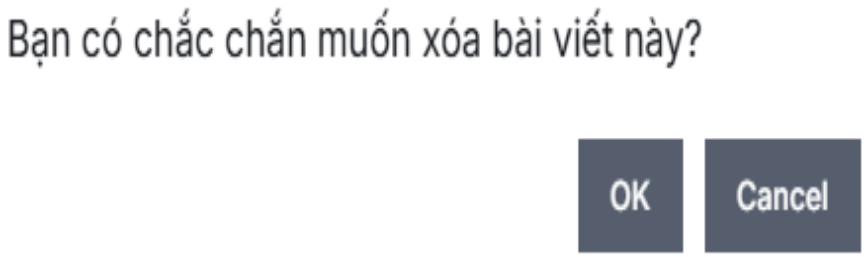


Figure 129: Delete blog page of DineGO System

3.10.11 Report Manager

3.10.11.1 Report List

Function trigger:

Triggered when the administrator selects **Guest Reports** from the sidebar menu.

URL: https://admin.dinego.shop/Report

Function description:

Displays a list of all user-submitted reports, including detail such as reporter, reported user, type, content, related links, status, report date, admin handling, and notes.

Normal execution case:

System loads the report list → admin can view all reports → admin may click **Edit** or **View Detail** for further actions.

Abnormal execution cases:

- No reports available → show an empty list message.
- Data load fails → show error message.

Screen layout:

- **Columns:** Reporter, Reported Guest, Type, Content, Related Links, Status, Report Date, Admin Handler, Admin Notes, Actions.
- **Buttons:** Edit, View Detail.

Danh sách báo cáo

ID	Người báo cáo	Người bị báo cáo	Loại	Nội dung	Liên kết liên quan	Trạng thái	Ngày tạo	Admin xử lý	Ghi chú admin	Hành động
1	Dev2 Id: (1)	2	1	99	0	Đã giải quyết	08/08/2025 00:00	Super Admin Id: (1)	N/A	
2	User123 (5)	User456 (8)	3	Nội dung không phù hợp	1	Đang chờ	15/07/2025 14:30	N/A	Đang chờ xem xét	
3	GuestUser (10)	AdminBot (99)	2	Spam bình luận	0	Đang xem xét	01/08/2025 09:00	Moderator Id: (5)	Đã chuyển cho nhóm chuyên trách	
4	Dev3 Id: (2)	10	1	Báo cáo sai phạm	0	Đã giải quyết	20/08/2025 11:15	Super Admin Id: (1)	Đã xử lý và cảnh báo	
5	Moderator (6)	User789 (12)	4	Ngôn ngữ xúc phạm	2	Đang chờ	25/08/2025 16:45	N/A	N/A	

Figure 130: Report manager page of DineGO System

3.10.11.2 Delete Report

Function trigger:

Triggered when the administrator clicks the **Delete** button for a report in the Report List page.

Function description:

Displays a confirmation popup asking the administrator to confirm deletion of the selected report.

Normal execution case:

Admin clicks **Delete** → confirmation popup appears → admin clicks **OK** → system deletes the report → updated list is displayed without the deleted entry.

Abnormal execution cases:

- Report not found → show error message.
- Deletion fails due to system error → show error message.

Screen layout:

- **Popup message:** “Are you sure you want to delete this report?”
- **Buttons:** OK, Cancel.

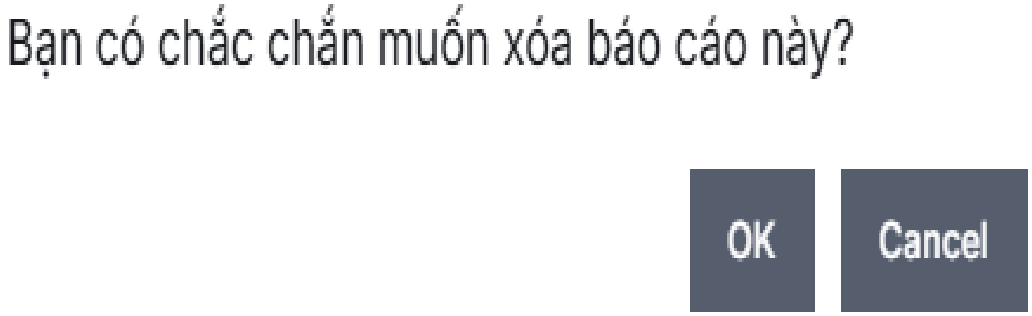


Figure 131: Delete report page of DineGO System

3.10.12 Contact With Customer

3.10.12.1 Contact List

Function trigger:

Triggered when the administrator selects **Contact with Customers** from the sidebar menu.

URL: https://admin.dinego.shop/Contact

Function description:

Displays a list of all customer contact requests, including sender detail, message content, admin replies, and status.

Normal execution case:

System loads the contact list → admin can view all contact messages → admin may click **Edit** or **Reply** to a message.

Abnormal execution cases:

- No contact messages available → show an empty list message.
- Data load fails → show error message.

Screen layout:

- **Columns:** ID, Full Name, Email, Title, Content, Status, Admin Reply, Reply Content, Date Sent, Date Replied, Actions.
- **Buttons:** Edit, Delete.

Danh sách Liên hệ

ID	Họ tên	Email	Tiêu đề	Nội dung	Trạng thái	Admin trả lời	Nội dung trả lời	Ngày gửi	Ngày trả lời	Hành động
3	Nguyễn Văn A	nguyen.van.a@gmail.com	Hỗ trợ kỹ thuật	Thiết bị của tôi không hoạt động đúng cách sau bản cập nhật.	Đã tiếp nhận	Đã phản hồi	Chào bạn, chúng tôi đã nhận được yêu cầu hỗ trợ kỹ thuật của bạn và sẽ liên hệ lại trong vòng 24 giờ tới để khắc phục sự cố.	08/08/2025 10:30	08/08/2025 11:00	
4	Trần Thị B	tran.thi.b@outlook.com	Yêu cầu thay đổi thông tin tài khoản	Tôi muốn cập nhật địa chỉ email và số điện thoại trên hồ sơ của mình.	Đang xử lý	Chưa phản hồi	Đang chờ bộ phận hỗ trợ thông tin khách hàng xử lý.	07/08/2025 14:00	N/A	
5	Lê Văn C	le.van.c@company.com	Phản hồi về chất lượng dịch vụ	Dịch vụ khách hàng của bạn rất tuyệt vời, tôi rất hài lòng với sự hỗ trợ.	Đã hoàn tất	Đã phản hồi	Cảm ơn bạn rất nhiều vì những lời khen ngợi. Chúng tôi rất vui được phục vụ bạn!	06/08/2025 09:15	06/08/2025 09:30	
6	Phạm Thu D	pham.thu.d@yahoo.com	Hóa đơn thanh toán	Tôi chưa nhận được hóa đơn cho tháng này.	Chưa giải quyết	Chưa phản hồi		05/08/2025 16:45	N/A	

Figure 132: Contact with customer manager page of DineGO System

3.10.12.2 Delete Contact

Function trigger:

Triggered when the administrator clicks the **Delete** button for a contact message in the Contact List page.

Function description:

Displays a confirmation popup asking the administrator to confirm deletion of the selected contact message.

Normal execution case:

Admin clicks **Delete** → confirmation popup appears → admin clicks **OK** → system deletes the contact message → updated list is displayed without the deleted entry.

Abnormal execution cases:

- Contact message not found → show error message.
- Deletion fails due to system error → show error message.

Screen layout:

- **Popup message:** "Are you sure you want to delete this contact?"
- **Buttons:** OK, Cancel.

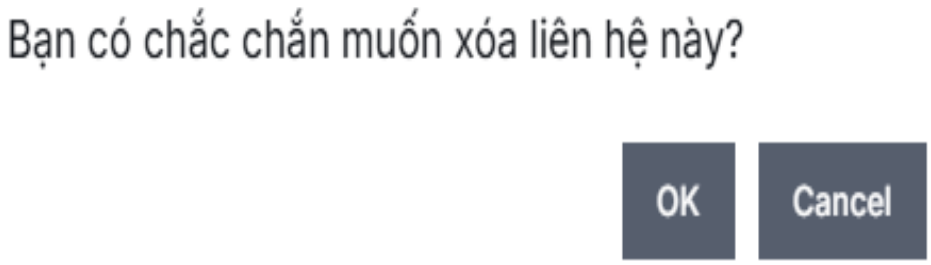


Figure 133: Delete contact with customer page of DineGO System

3.10.13 System Log

Function trigger:

Triggered when the administrator selects **System Log** from the sidebar menu.

URL: https://admin.dinego.shop/SystemLog

Function description:

Displays a list of all recorded system activities performed by administrators, including actions, descriptions, timestamps, IP addresses, devices, and status codes.

Normal execution case:

System loads the system log list → admin can view recent activities → admin may filter or sort logs to find specific events.

Abnormal execution cases:

- No logs available → show an empty list message.
- Data load fails → show error message.

Screen layout:

- **Columns:** ID, Admin, Action, Description, Time, IP, Device, Status Code, Success Status.

Nhật ký hệ thống

ID	Admin	Hành động	Mô tả	Thời gian	IP	Thiết bị	Mã trạng thái	Thành công
39	Super Admin ID: 1	POST	Thêm Blog	11/08/2025 11:30:28	::1	Mozilla/5.0 (Windows NT 10.0; Win64; x64)	200	✓
38	Super Admin ID: 1	POST	Thêm Blog	11/08/2025 11:30:21	::1	Mozilla/5.0 (Windows NT 10.0; Win64; x64)	200	✓
37	Super Admin ID: 1	POST	Thực hiện/Auth/Login	11/08/2025 11:25:04	::1	Mozilla/5.0 (Windows NT 10.0; Win64; x64)	200	✓
36	Super Admin ID: 1	GET	Xem trang/dashboard	11/08/2025 11:20:00	192.168.1.10	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)	200	✓
35	Super Admin ID: 1	PUT	Cập nhật hồ sơ	11/08/2025 11:15:30	::1	Mozilla/5.0 (Windows NT 10.0; Win64; x64)	200	✓
34	Admin Khách ID: 5	DELETE	Xóa tệp/files/report.pdf	11/08/2025 11:10:05	203.0.113.45	Mozilla/5.0 (Linux; Android 10; SM-G960F)	403	X

Figure 134: System log page of DineGO System

3.10.14 Update Avatar

Function trigger: Triggered when the user clicks the avatar image in their profile section to change their avatar.

Function description: Displays a popup containing a selection of avatar images for the user to choose from.

The user can click on any avatar to select it, and the system updates their profile picture accordingly.

Normal execution case: Guest clicks Change Avatar → popup appears with available avatars → user clicks on desired avatar → system updates avatar → popup closes.

Abnormal execution cases: Avatar selection fails due to server error → show error message. Guest closes the popup without selecting → no changes are made.

Screen layout: Avatar grid: Multiple selectable avatar images arranged in rows and columns.

Buttons: Close.

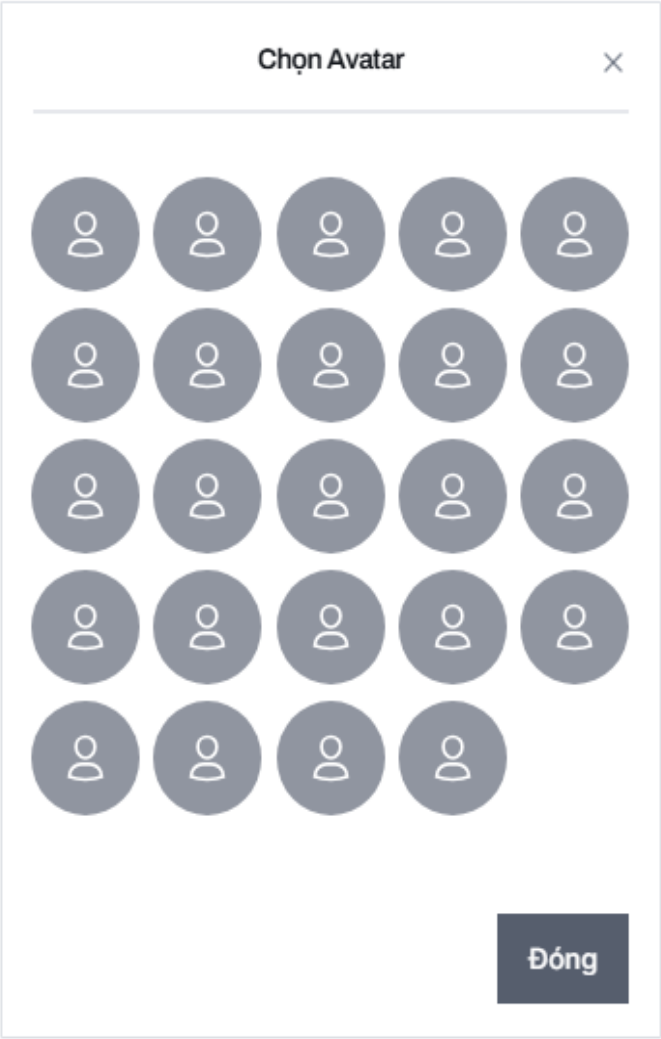


Figure 135: Avatar update in admin page.

4. Non-Functional Requirements

4.1 Usability

- **USA-1:** The system shall provide a user-friendly and intuitive interface, compatible.
- **USA-2:** The user interface shall be designed with clear navigation, consistent layout, and responsive design to ensure good experience across screen sizes.
- **USA-3:** The system shall provide tooltips, error hints, and confirmation messages to guide users during interactions.
- **USA-4:** New users should be able to perform basic tasks (register, book table, order food) within 5 minutes of first use without training.
- **USA-5:** All public pages must follow WCAG 2.1 Level AA standards for accessibility where feasible.

4.2 Reliability

- **REL-1**: The system shall be available 99% of the time, excluding planned maintenance.

- **REL-2**: The system shall have a Mean Time Between Failures (MTBF) of at least 500 hours.
- **REL-3**: The system shall have a Mean Time To Repair (MTTR) of less than 4 hours in case of critical failures.
- **REL-4**: Automated failover mechanisms and data backup shall be implemented daily to prevent data loss.
- **REL-5**: Error handling must ensure that user data is not lost in the event of unexpected failures.

4.3 Performance

- **PER-1:** Average page load time shall be under 3 seconds for 95% of requests under normal network conditions.
- **PER-2**: The system shall handle at least 100 concurrent users performing booking, ordering, or browsing without significant degradation in response time.
- **PER-3**: The system shall process payment transactions within 5 seconds on average.
- **PER-4**: Search and filter operations shall return results within 2 seconds for queries with a dataset of up to 10,000 records.
- **PER-5**: The system database shall be optimized to maintain response times for CRUD operations below 1 second for 90% of transactions.

4.4 Security

- **SEC-1**: Guest passwords must be stored using hashed and salted encryption algorithms (e.g., bcrypt).
- **SEC-2**: All data in transit shall be encrypted using HTTPS/TLS.
- **SEC-3**: Role-based access control (RBAC) must be enforced to ensure users only access functions and data permitted for their role.
- **SEC-4**: Session management must ensure automatic logout after 30 minutes of inactivity.
- **SEC-5**: Regular security audits and vulnerability scans shall be performed at least once per quarter.
- **SEC-6**: The system shall comply with relevant data privacy regulations (e.g., local Personal Data Protection regulations).

5. Requirement Appendix

5.1 Common Requirements

- **CR-01**: The system shall support modern web browsers including Google Chrome, Microsoft Edge, and Mozilla Firefox (latest two versions).
- **CR-02**: The user interface must be responsive and function properly across desktops, tablets, and mobile devices.
- **CR-03**: The default interface language is Vietnamese. The system should support internationalization (i18n) for future multilingual expansion.
- **CR-04**: All input fields must implement both client-side and server-side validation (e.g., required fields, format validation, max-length constraints).
- **CR-05**: Guest sessions shall automatically expire after 30 minutes of inactivity. Sensitive actions must require re-authentication.
- **CR-06**: All key features must be accessible within a maximum of three clicks from the homepage.
- **CR-07**: The application shall follow the MVC (Model-View-Controller) architecture to maintain separation of concerns.
- **CR-08**: All forms shall provide clear validation messages and feedback upon success or error.
- **CR-09**: The system must log critical user actions such as login, reservations, orders, and reviews for auditing purposes.

5.2 Application Messages List

This section defines the standard messages displayed by the DineGO system to users in various contexts. Each message includes a unique message code, the type of message (e.g., inline, toast), the context in which it appears, and the exact content shown to the user. These messages ensure consistent communication across the application, improve usability, and provide clear feedback or guidance to users during interactions.

Table 17*: List of Application Messages*

#	Message code	Message Type	Context	Content
1	MSG01	In line	There is not any search result	*No search results.*
2	MSG02	In red, under the text box	Input-required fields are empty	*The * field is required.*
3	MSG03	Toast message	Updating asset(s) information successfully	*Update asset(s) successfully.*
4	MSG04	Toast message	Adding new asset successfully	*Add assets successfully.*
5	MSG05	Toast message	Confirming email of asset hand-over is sent successfully	*A confirmation email has been sent to {email_address}.*
6	MSG06	Toast message	Resetting asset information successfully	*Return asset(s) successfully.*
7	MSG07	Toast message	Deleting asset information successfully	*Delete asset(s) successfully.*
8	MSG08	In red, under the text box	Input value length > max length	*Exceed max length of {max_length}.*
9	MSG09	In line	Guest name or password is not correct when clicking sign-in	*Incorrect username or password. Please check again.*

5.3 Other Requirements.

- **OR-01**: The system shall integrate with third-party APIs such as Google Maps for location services and MoMo for digital payments.
- **OR-02:** All restaurant license documents must be securely stored and accessible by administrators for verification purposes.
- **OR-03**: All source code must follow C# coding conventions and include XML documentation for all public methods.
- **OR-04**: The database must be backed up automatically on a daily basis to VPS or secure cloud storage.
- **OR-05**: Vouchers must support condition-based rules such as applicable categories or minimum order amounts.
- **OR-06**: Loyalty points must be automatically updated upon successful payments and displayed in the customer dashboard.

- **OR-07**: All system-generated emails (e.g., booking confirmation, password reset) must be sent via authenticated SMTP with retry logic.
- **OR-08**: The admin dashboard must support CSV/PDF export functions for reports such as revenue, user logs, and license approvals.

5.4 Training Dataset & Strategy.

5.4.1 Tag Prediction (DistilBERT)

Training Dataset:

- Source file: food_descriptions_20k.csv
- Each row contains:
    - text: user-style food description
    - label: multiple tags (comma-separated)
- Tags encoding: binary vectors using labels.json (~30 tags, e.g., mon_ngot, mon_sang, mon_chien, mon_nhat, etc.)

Training Strategy

- Model: DistilBERTForSequenceClassification
- Task: Multi-label classification
- Loss Function: BCEWithLogitsLoss
- Tokenizer: DistilBertTokenizerFast
- Epochs: 3
- Batch Size: 8 (with gradient accumulation)
- Prediction Threshold: 0.3
- Objective: Light fine-tuning on domain-specific dataset without losing general language knowledge from the base model.

5.4.2 Response Generation (VIT5)

Training Dataset:

- Source file: prompt_response_15k_concrete.csv
- Each row contains:
    - prompt: user context or intent (in Vietnamese)
      target: descriptive food suggestion
- Dataset size: ~15,000 samples

Training Strategy:

- Model: VietAI/vit5-base
  Task: Text generation (sequence-to-sequence)
- Loss Function: Cross-Entropy
- Tokenizer: AutoTokenizer
- Epochs: 5
- Batch Size: 4 (gradient accumulation = 2)
  Max Sequence Length: 64
- Learning Rate: 5e-5
- Objective: Fine-tune on curated Vietnamese dataset for contextual and descriptive food suggestions.

IV. Software Design Description

1. Overall Description

1.1 Assumptions

The DineGO system is developed based on the following assumptions:

- Operating Systems: Windows 10, 11; macOS Monterey or newer
- Supported Browsers: Google Chrome, Microsoft Edge, Mozilla Firefox
- Frontend Framework: ASP.NET MVC 5.0
- Backend Platform: ASP.NET API Core 5.0
- Database: Microsoft SQL Server 2022
- Development Tools:
    - Visual Studio Code
    - SSMS (SQL Server Management Studio)
    - GitHub for version control
- Hosting: VPS or similar cloud-based virtual servers
- Payment Integration: VNPay Sandbox environment
- AI Module (optional)
- Collaboration: Microsoft Teams / Google Meet

1.2 Design Constraints

The system must comply with the following design and operational constraints:

- End-users are expected to access the system using modern browsers on compatible OSs.
- The system must provide responsive design to support both desktop and mobile devices.
- Security must be ensured through:
    - Password encryption (bcrypt or equivalent)Session management (cookie expiration after 2 hours)
    - Google login integration for quick and secure access
- Database transactions must ensure ACID properties for order, reservation, and payment operations.
- All restaurant verification documents (images, licenses) must be stored securely on cloud storage (e.g., AWS S3).
- System updates (feature or bug-fixes) will be deployed periodically upon stakeholder agreement and sprint reviews.
- Unit testing, integration testing, system testing, and user acceptance testing (UAT) must be conducted before any major release.
- Reservation confirmation must be done within 15 minutes to prevent auto-cancellation.
- Only verified restaurants can access full management features such as post publishing and voucher generation.

1.3 Technology Suggestion

For long-term maintainability and scalability, the following technologies are recommended:

- **Frontend**: ASP.NET MVC 5.0 for a component-based UI and modern JavaScript ecosystem
- **Backend**: ASP.NET API Core 5.0 for building secure RESTful APIs with high performance
- **Database**: SQL Server with Entity Framework Core for ORM mapping and efficient queries
- **Real-Time Communication (future option)**: SignalR + Redis Cache (if real-time order tracking or chat is needed)
- **Cloud Storage**: VPS for storing restaurant images, videos, and documents
- **Authentication**: OAuth2.0 integration via Google Sign-In for secure and quick login
- **Monitoring**: Admin dashboard with system logs and activity tracking (stored in system_log table.

2. System Architecture Design

2.1 Overall Architecture

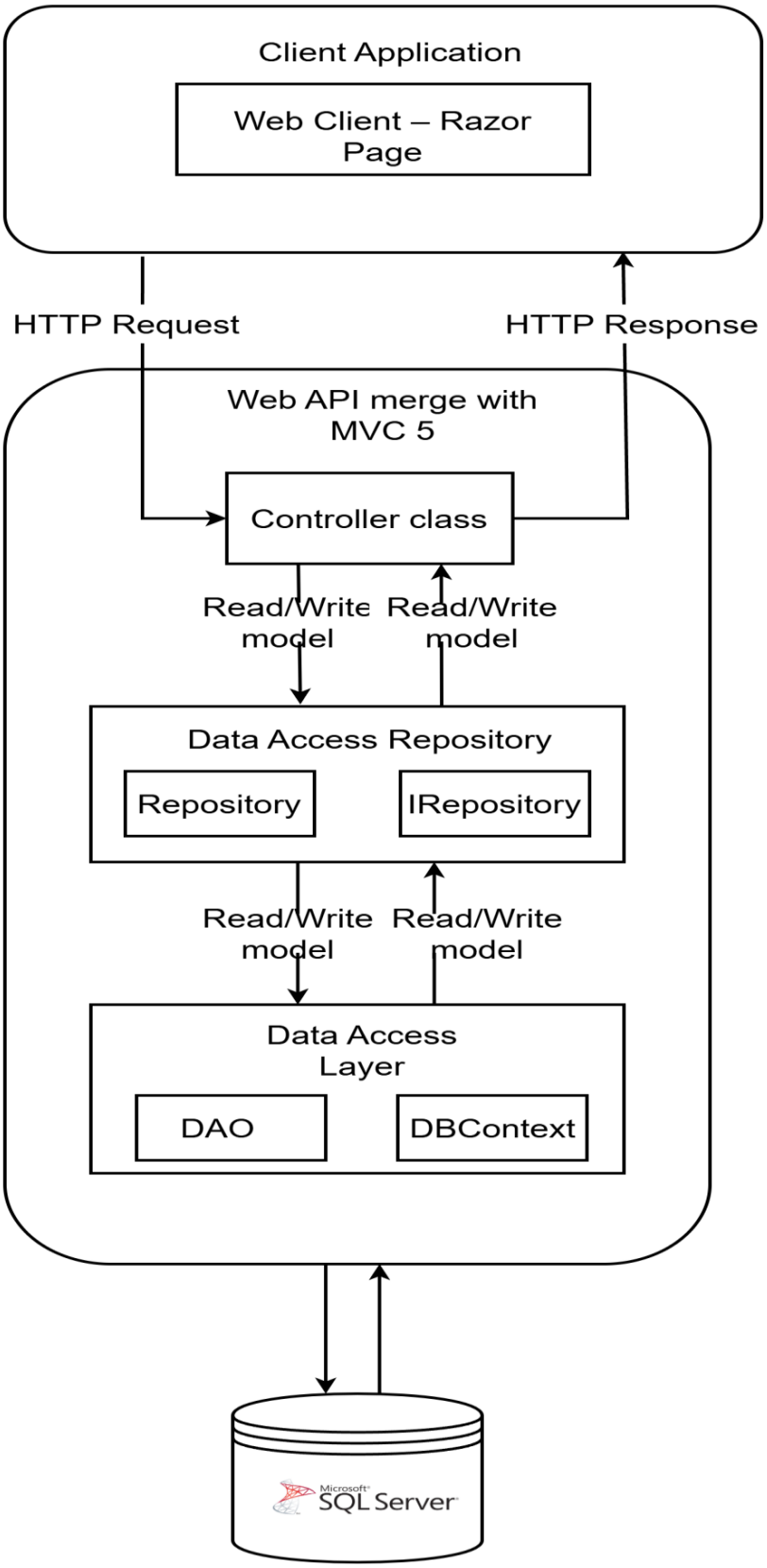


Figure 136: Overall Architecture Design

2.2 System Architecture

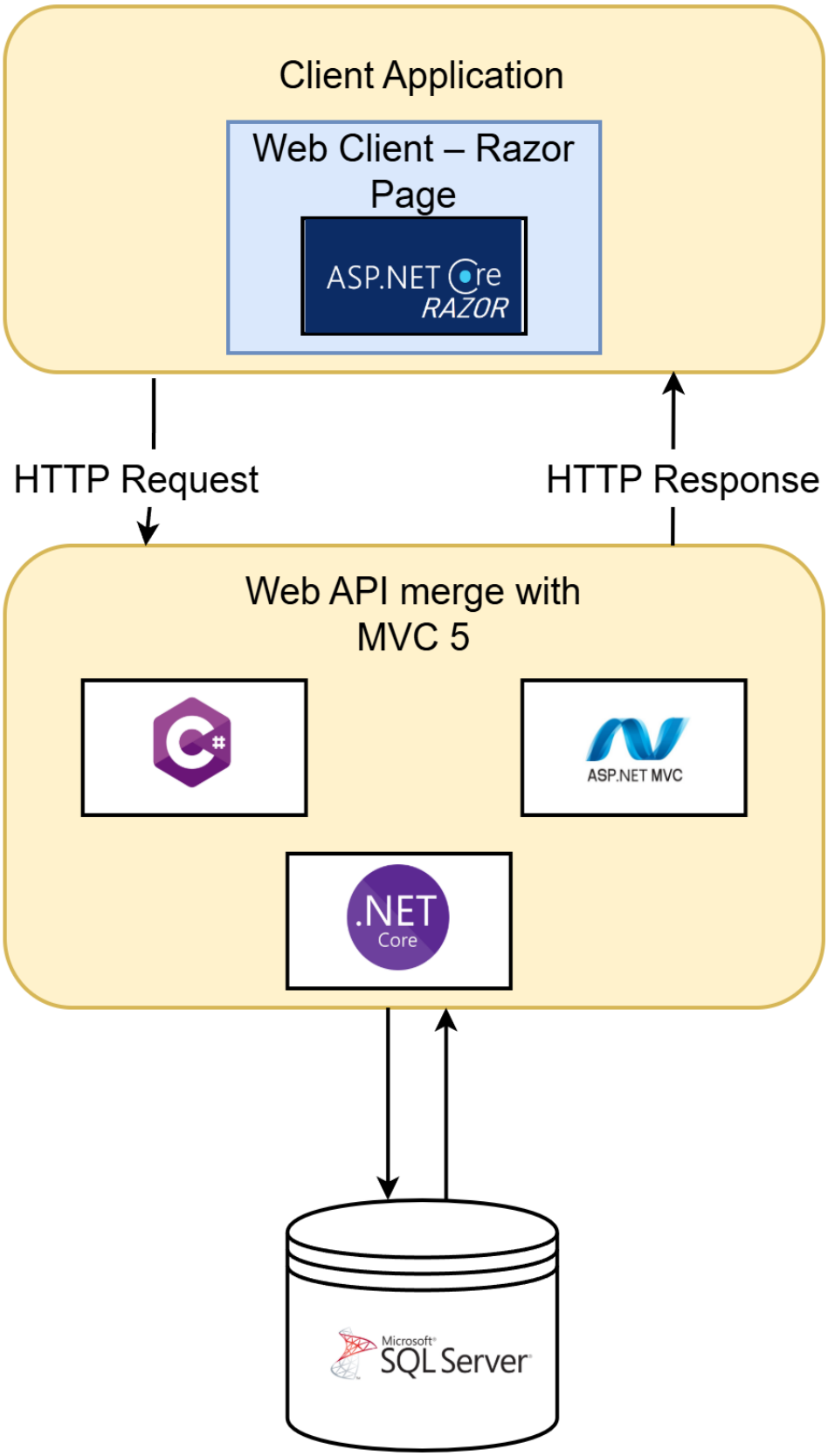


Figure 137: System Architecture

2.3 Package Diagram

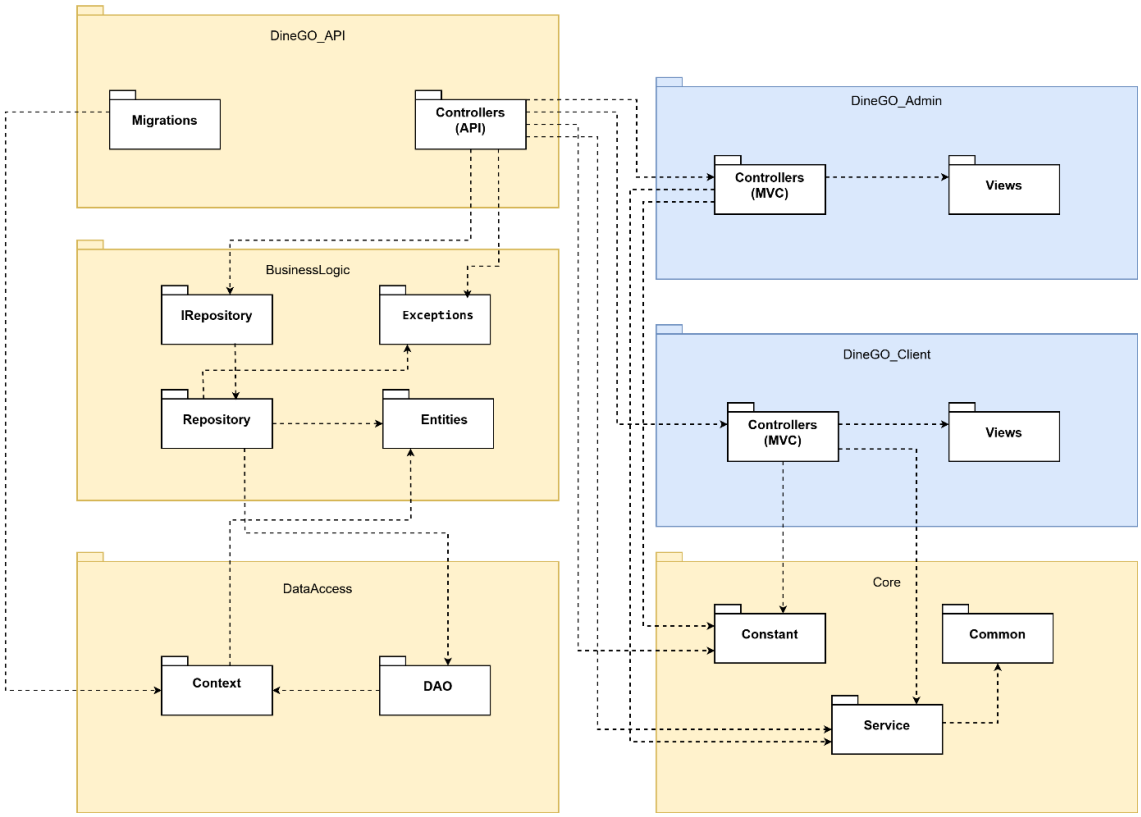


Figure 138: Package Diagram

Package Descriptions

This table lists the main packages in the DineGO project along with their respective purposes, providing an overview of their roles within the system architecture

Table 18 :Package Descriptions

No	Package	Description
01	Controllers (API)	Contains API controllers handling HTTP requests.
02	Controllers (MVC)	Contains MVC controllers for Razor view rendering.
03	Migrations	Contains Entity Framework Core migration files.
04	Views	Contains Razor view pages for rendering the user interface.
05	IRepository	Contains interfaces for repository classes.
06	Repository	Contains implementation of the repository pattern.
07	Entities	Contains entity classes that map to database tables.
08	Exceptions	Contains custom exception classes for error handling.
09	Context	Contains the database context class for Entity Framework.
10	DAO	Contains classes for specific data access operations.
11	Constant	Contains project-wide constant definitions.
12	Common	Contains shared config and utility classes.
13	Service	Contains services that call API endpoints and return data to the MVC layer.

2.4 Common Services

This section describes common cross-cutting services shared across the DineGO system:

1. Cookie Consent Management

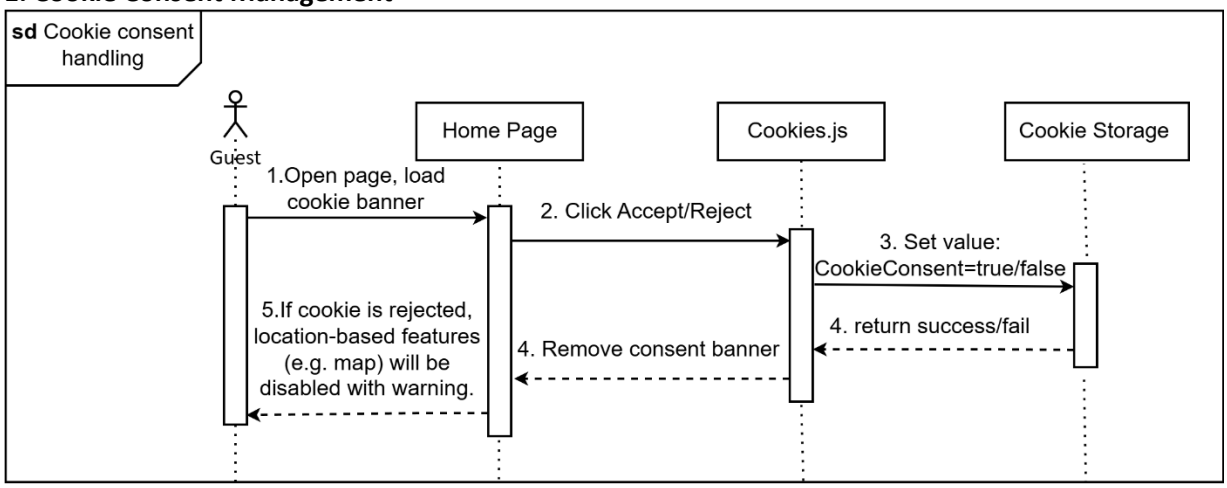


Figure 139: sequence diagram cookie consent handling (Guest)

A lightweight JavaScript-based cookie management module (Cookies.js) is used to obtain customer consent for using browser cookies.

- The customer 's choice is stored in the CookieConsent cookie with a 1-year expiry.
- Location-related features (e.g., map modal, geolocation, address picker) require consent to proceed.
- If the customer does not accept cookies, features like map interaction will be disabled via alert and modal closure.

2. Session-based Customer Location

After location selection (manual or via map), the location is saved in HttpContext.Session["USER_LOCATION"] via /Location/SetUserLocation.

This session data is used to:

- Filter restaurants and food items by proximity
- Pre-fill address fields
- Support personalized recommendations in the future

3. Map-based Location Picker (Profile / Address Entry)

This module supports choosing a precise location (for personal profile or restaurant address) using Leaflet. It performs:

- Reverse geocoding (get ward/district/province)
- Set hidden fields (cus_latitude, cus_longitude, cus_address)
- Update DOM (#selectedLocation)
- Used in locationPicker.js
- No restaurant marker, no distance calculation

4. Nearby Restaurant Finder (Food.js – Distance & Marker Display)
This module is used in the food listing view to enhance customer experience with location-based filtering. It performs the following:

- Load restaurant markers from the API endpoint /food/markers, each containing name, address, coordinates, and image.
- Get customer location from session via HttpContext.Session["USER_LOCATION"], which is set when the customer selects a location using the map modal.
- Calculate the distance between each restaurant and the user's location using the Haversine formula (via GeoHelper.CalculateDistanceKm(...)).

$$Distance = 2R \cdot artan2(\sqrt{a}, \sqrt{1-a})$$

- Sort restaurants by calculated proximity before displaying them to prioritize nearby options.
- Fallback handling: If no GPS access or customer location is set, the feature is gracefully skipped.

5. Utility Scripts
Additional scripts such as GeoHelper.cs and locationPicker.js support reverse geocoding, map interaction, and address parsing using OpenStreetMap APIs.
Disclaimer: The map interface used in this application is for illustrative purposes only and is based on public map sources such as OpenStreetMap. ***It does not represent any political or territorial claims***. This precaution is especially relevant as some third-party maps may include disputed features (e.g., the nine-dash line in the South China Sea).
These components work in tandem with cookies and session logic.

2.5 AWS Service (S3 Bucket)

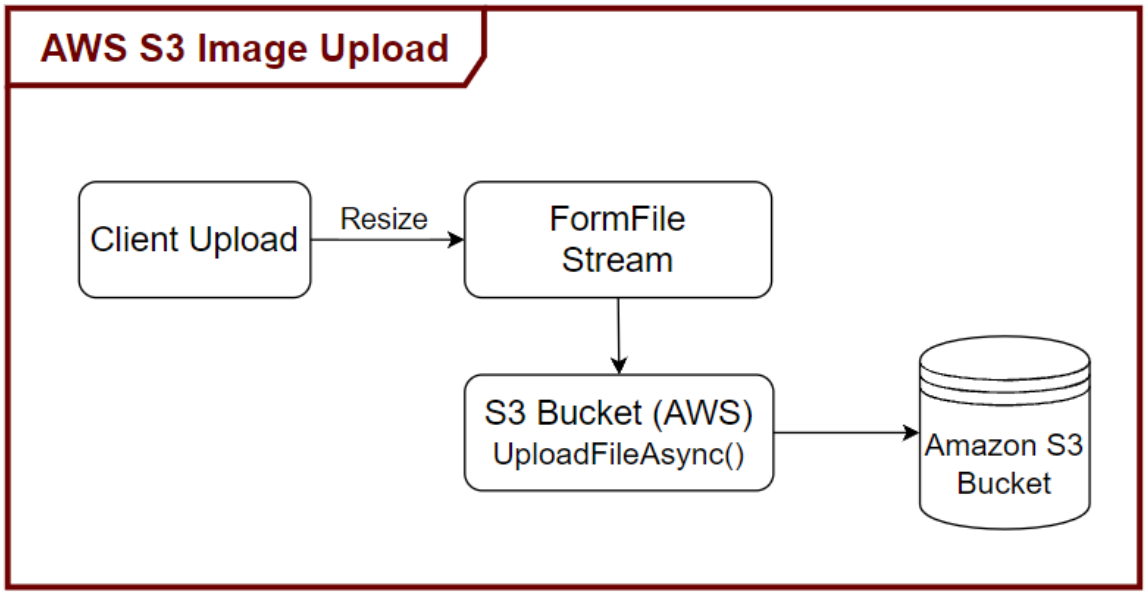


Figure 140: AWS S3 Image Upload.

Purpose:

AWS S3 (Amazon Simple Storage Service) is used to store media files such as restaurant images, food photos, avatars, and post attachments. It provides a scalable and highly durable storage infrastructure that supports secure upload and retrieval operations from our web application.

Integration Architecture:
The application integrates with AWS S3 through a utility class S3BucketAWS.cs, and image processing is handled by ImageHelper.cs.
Flow of image upload:

- The user uploads an image (e.g., post/restaurant…).
- ImageHelper resizes the image if necessary:
- Resize to 1920px max for full image.
- Resize to predefined width for thumbnail.
- Both versions (full_ and thumb_) are uploaded to S3 using UploadFileAsync(...) in S3BucketAWS.
- File names follow this pattern: folder/full_<filename>.jpg and folder/thumb_<filename>.jpg

Flow of image deletion:

- When the user updates or deletes an entity (e.g., post or restaurant), the corresponding S3 objects (full_ and thumb_) are deleted by key.

Security and Configuration:
AWS credentials and bucket detail are stored in appsettings.json (or in environment variables in production):

```
"AWS": {
  "AccessKey": "...",
  "SecretKey": "...",
  "Region": "ap-southeast-1",
  "BucketName": "dinego-bucket-aws"
}
```

Figure 141: AWS credentials and bucket detail are stored in appsettings.json

AccessKey / SecretKey: Used to authenticate the app with AWS S3.
Region: The S3 bucket's deployment region.
BucketName: Name of the bucket storing images.
Service Classes Used:
S3BucketAWS.cs: Encapsulates logic to upload and delete files from AWS S3.
ImageHelper.cs: Resizes uploaded images and prepares thumbnails before calling the S3 service.
Technologies Used:
AWS SDK for .NET: Provides the AmazonS3Client, TransferUtility, and request models.
System.Drawing: Used to resize images on server before upload.
.NET Core Configuration: Injects and uses appsettings for credentials.

2.6 VietQR

VietQR is integrated into the system as a fast payment QR code solution based on the **Napas 247** standard, allowing customers to scan the QR code via Mobile Banking applications or e-wallets to make quick and secure bank transfers.

How it works in the DineGO system:

- When a customer creates a reservation, the system generates a **dynamic VietQR code** containing account number, account holder name, amount, and transaction description.
- The QR code is displayed directly on the payment interface, along with a 10-minute countdown timer.
- The system periodically calls the Casso API to verify payment transactions that match the required detail(amount, description, timestamp).
- If the payment is valid, the reservation status is updated to “Confirmed”; if the payment times out, the reservation is automatically canceled.

Technologies Used:

- VietQR API to generate dynamic payment QR codes.
- Casso API to retrieve and validate transaction data.
- .NET Core MVC to display the QR code and manage the payment workflow.
- JavaScript (payment.js) to handle countdown timers and periodic payment checks.

Benefits:

- Minimizes errors from manual data entry.
- Fast, secure, and compatible with most banks in Vietnam.
- Seamlessly integrated with DineGO’s reservation management system.

2.7 Database Design

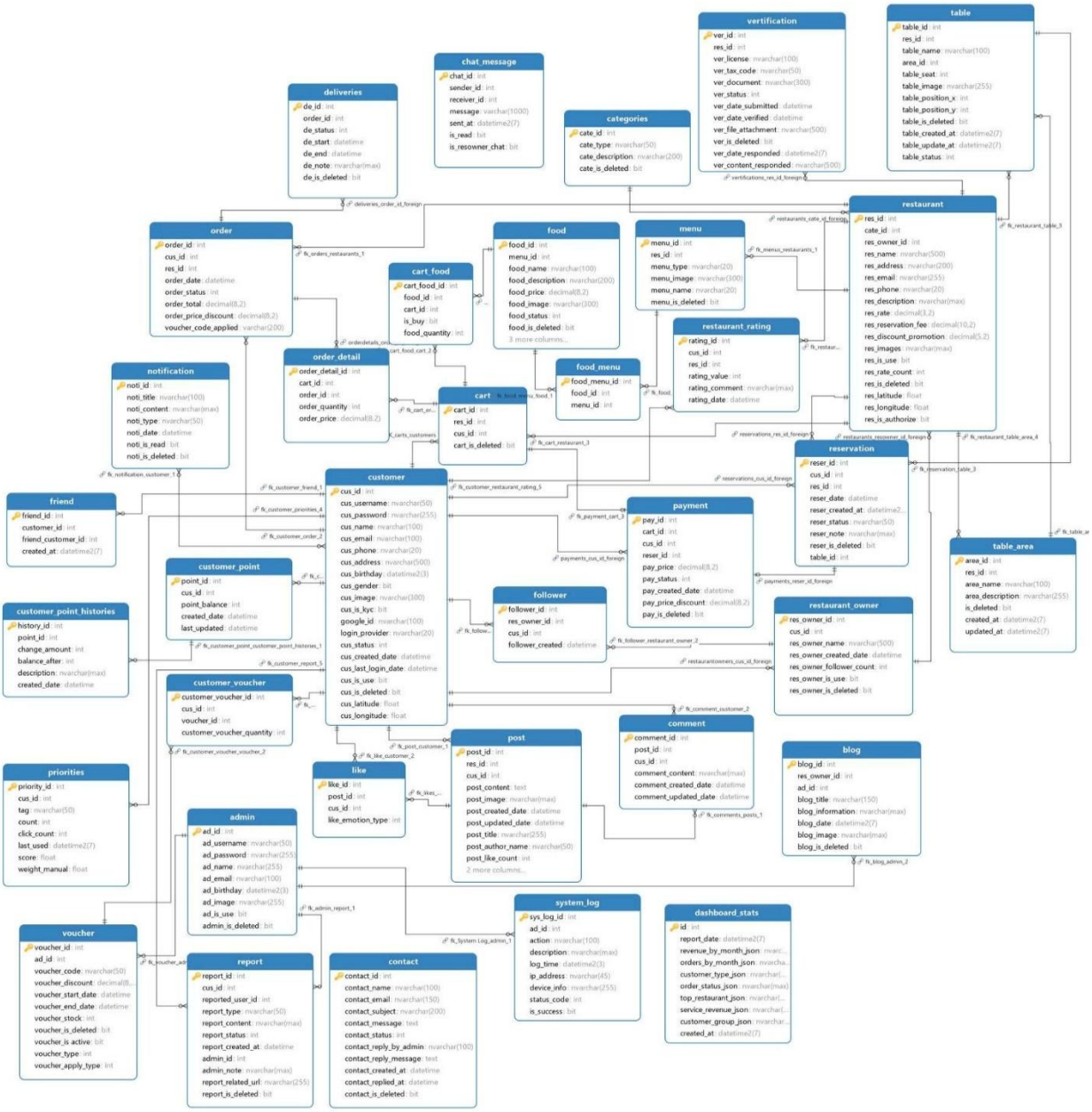


Figure 142: Database Design [1]

Table Descriptions

a. admin table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	ad_id	int		Yes	Yes	PK	Primary key
2	ad_username	nvarchar	50	No	Yes		
3	ad_password	nvarchar	255	No	Yes		
4	ad_name	nvarchar	255	No	Yes		
5	ad_email	nvarchar	100	No	Yes		
6	ad_birthday	datetime2(3)		No	Yes		
7	ad_image	nvarchar	255	No	No		Nullable
8	ad_is_use	bit		No	No		Nullable
9	admin_is_deleted	bit		No	Yes		

b. customer table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	cus_id	int		Yes	Yes	PK	Primary key
2	cus_username	nvarchar	50	No	Yes		
3	cus_password	nvarchar	255	No	Yes		
4	cus_name	nvarchar	100	No	Yes		
5	cus_email	nvarchar	100	No	Yes		
6	cus_phone	nvarchar	20	No	Yes		
7	cus_address	nvarchar	500	No	No		
8	cus_birthday	datetime2(3)		No	No		Nullable
9	cus_gender	bit		No	No		Nullable
10	cus_image	nvarchar	300	No	No		Nullable
11	cus_is_kyi	bit		No	No		Nullable

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
12	google_id	nvarchar	100	No	No		Nullable
13	login_provider	nvarchar	20	No	No		Nullable
14	cus_status	int		No	No		Nullable
15	cus_created_date	datetime		No	No		Nullable
16	cus_last_login_date	datetime		No	No		Nullable
17	cus_is_use	bit		No	Yes		Default = 1
18	cus_is_deleted	bit		No	Yes		
19	cus_latitude	float		No	Yes		
20	cus_longitude	float		No	Yes		

c. *restaurant_owner table*

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	res_owner_id	int		Yes	Yes	PK	Primary key
2	cus_id	int		No	Yes	PK	FK →customer.cus_id
3	res_owner_name	nvarchar	500	No	Yes		
4	res_owner_created_date	datetime		No	Yes		
5	res_owner_follower_count	int		No	Yes		
6	res_owner_is_use	bit		No	Yes		
7	res_owner_is_deleted	bit		No	Yes		

d. *restaurant table*

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	res_id	int		Yes	Yes	PK	Primary key
2	cate_id	int		No	Yes	FK	FK → category.cate_id
3	res_owner_id	int		No	Yes	FK	FK → restaurant_owner.res_owner_id
4	res_name	nvarchar	500	No	Yes		
5	res_address	nvarchar	200	No	No		Nullable
6	res_email	nvarchar	255	No	No		Nullable
7	res_phone	nvarchar	20	No	No		Nullable
8	res_description	nvarchar(max)		No	No		Nullable
9	res_rate	decimal(3,2)		No	No		Nullable
10	res_reservation_fee	decimal(10,2)		No	No		Nullable
11	res_discount_promotion	decimal(5,2)		No	No		Nullable
12	res_images	nvarchar(max)		No	No		Nullable
13	res_is_use	bit		No	Yes		Default = 1
14	res_rate_count	int		No	Yes		
15	res_is_deleted	bit		No	Yes		
16	res_latitude	float		No	No		Nullable
17	res_longitude	float		No	No		Nullable
18	res_is_authorize	bit		No	No		

e. *category table*

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	cate_id	int		Yes	Yes	PK	Primary key
2	cate_type	nvarchar	50	No	Yes		

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
3	cate_description	nvarchar	200	No	No		Nullable
4	cate_is_deleted	bit		No	Yes		

f. menu table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	menu_id	int		Yes	Yes	PK	Primary key
2	res_id	int		No	Yes	FK	FK → restaurant.res_id
3	menu_type	nvarchar	20	No	Yes		
4	menu_image	nvarchar	300	No	Yes		
5	menu_name	nvarchar	20	No	Yes		
6	menu_is_deleted	bit		No	Yes		

g. food table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	food_id	int		Yes	Yes	PK	Primary key
2	menu_id	int		No	Yes	FK	FK → menu.menu_id
3	food_name	nvarchar	100	No	Yes		
4	food_description	nvarchar	200	No	Yes		
5	food_price	decimal(8,2)		No	Yes		
6	food_image	nvarchar	300	No	Yes		
7	food_status	int		No	Yes		
8	food_is_deleted	bit		No	No		Nullable
9	food_tag	nvarchar	200	No	No		

h. food_menu table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	food_menu_id	int		Yes	Yes	PK	Primary key

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
2	food_id	int		No	No	FK	FK → food.food_id
3	menu_id	int	20	No	No	FK	FK → menu.menu_id

i. cart table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	cart_id	int		Yes	Yes	PK	Primary key
2	res_id	int		No	No	FK	FK → restaurant.res_id
3	cus_id	int		No	No	FK	FK → customer.cus_id
4	cart_is_deleted	bit		No	Yes		

j. cart_food table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	cart_food_id	int		Yes	Yes	PK	Primary key
2	food_id	int		No	No	FK	FK → food.food_id
3	cart_id	int		No	No	FK	FK → cart.cart_id
4	is_buy	bit		No	No		
5	food_quantity	int		No	No		

k. order table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	order_id	int		Yes	Yes	PK	Primary key
2	cus_id	int		No	Yes	FK	FK → customer.cus_id
3	res_id	int		No	Yes	FK	FK → restaurant.res_id
4	order_date	datetime		No	Yes		
5	order_status	int		No	Yes		
6	order_total	decimal(8,2)		No	Yes		
7	order_price_discount	decimal(8,2)		No	No		Nullable
8	voucher_code_applied	varchar	200	No	No		

l. order_detail table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	order_detail_id	int		Yes	Yes	PK	Primary key
2	cart_id	int		No	No	FK	FK → cart.cart_id
3	order_id	int		No	Yes	FK	FK → order.order_id
4	order_quantity	int		No	Yes		
5	order_price	decimal(8,2)		No	Yes		

m. reservation table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	reser_id	int		Yes	Yes	PK	Primary key
2	cus_id	int		No	Yes	FK	FK → customer.cus_id

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
3	res_id	int		No	Yes	FK	FK → restaurant.res_id
4	table_id	int		No	Yes	FK	FK → table.table_id
5	reser_date	datetime		No	Yes		
6	reser_created_at	datetime2(7)		No	Yes		
7	reser_status	nvarchar	50	No	Yes		
8	reser_note	nvarchar(max)		No	No		
9	reser_is_deleted	bit		No	Yes		

n. payment table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	pay_id	int		Yes	Yes	PK	Primary key
2	cart_id	int		No	No	FK	FK → cart.cart_id
3	cus_id	int		No	Yes	FK	FK → customer.cus_id
4	reser_id	int		No	No	FK	FK → reservation.reser_id
5	pay_price	decimal(8,2)		No	Yes		
6	pay_status	int		No	No		
7	pay_created_date	datetime		No	Yes		
8	pay_price_discount	decimal(8,2)		No	No		
9	pay_is_deleted	bit		No	Yes		

o. voucher table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	voucher_id	int		Yes	Yes	PK	Primary key

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
2	ad_id	int		No	No	FK	FK → admin.ad_id
3	voucher_code	nvarchar	50	No	Yes		
4	voucher_discount	decimal(8,2)		No	Yes		
5	voucher_start_date	datetime		No	Yes		
6	voucher_end_date	datetime		No	Yes		
7	voucher_stock	int		No	No		
8	voucher_is_deleted	bit		No	Yes		
9	voucher_is active	bit		No	No		
10	voucher_type	int		No	Yes		
11	voucher_apply_type	int		No	Yes		

p. customer_voucher table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	customer_voucher_id	int		Yes	Yes	PK	Primary key
2	cus_id	int		No	Yes	FK	FK → customer.cus_id
3	voucher_id	int		No	Yes	FK	FK → voucher.voucher_id
4	voucher_discount	int		No	Yes		

q. post table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	post_id	int		Yes	Yes	PK	Primary key
2	res_id	int		No	No	FK	FK → restaurant.res_id

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
3	cus_id	int		No	No	FK	FK → customer.cus_id
4	post_content	text		No	Yes		
5	post_image	nvarchar(max)		No	Yes		
6	post_created_date	datetime		No	Yes		
7	post_updated_date	datetime		No	Yes		
8	post_title	nvarchar	255	No	No		Nullable
9	post_author_name	nvarchar	50	No	No		Nullable
10	post_like_count	int		No	No		Nullable
11	post_comment_count	int		No	No		Nullable
12	post_is_approve	bit		No	No		

r. comment table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	comment_id	int		Yes	Yes	PK	Primary key
2	post_id	int		No	Yes	FK	FK → post.post_id
3	cus_id	int		No	Yes	FK	FK → customer.cus_id
4	comment_content	nvarchar(max)		No	Yes		
5	comment_created_date	datetime		No	Yes		
6	comment_updated_date	datetime		No	Yes		

s. *like table*

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	like_id	int		Yes	Yes	PK	Primary key
2	post_id	int		No	No	FK	FK → post.post_id
3	cus_id	int		No	No	FK	FK → customer.cus_id
4	like_emotion_type	int		No	No		

t. *blog table*

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	blog_id	int		No	Yes	PK	Primary key
2	res_owner_id	int		No	Yes	FK	FK → restaurant_owner.res_owner_id
4	ad_id	int		No	No		
5	blog_title	nvarchar	150	No	Yes		
6	blog_information	nvarchar(max)		No	Yes		
7	blog_date	datetime2(7)		No	Yes		
8	blog_image	nvarchar(max)		No	No		Nullable
9	blog_is_deleted	bit		No	Yes		

u. *follower table*

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	follower_id	int		Yes	Yes	PK	Primary key
2	res_owner_id	int		No	Yes	FK	FK → restaurant_owner.res_owner_id
3	cus_id	int		No	Yes	FK	FK → customer.cus_id

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
4	follower_created	datetime		No	Yes		

v. notification table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	noti_id	int		Yes	Yes	PK	Primary key
2	noti_title	nvarchar	100	No	Yes		
3	noti_content	nvarchar(max)		No	Yes		
4	noti_type	nvarchar	50	No	Yes		
5	noti_date	datetime		No	Yes		

w. notification_customer table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	noti_customer_id	int		Yes	Yes	PK	Primary key
2	noti_id	itn		No	Yes	FK	FK → notification.noti_id
3	cus_id	int		No	Yes	FK	FK → customer.cus_id
4	noti_is_read	bit		No	Yes		
5	noti_date	datetime		No	Yes		
6	noti_type	nvarchar	50	No	Yes		
7	noti_is_deleted	bit		No	Yes		

x. deliveries table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	de_id	int		Yes	Yes	PK	Primary key
2	order_id	int		No	Yes	FK	FK → order.order_id
3	de_status	int		No	Yes		

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
4	de_start	datetime		No	Yes		
5	de_end	datetime		No	Yes		
6	de_note	nvarchar(max)		No	No		
7	de_is_deleted	bit		No	Yes		

y. verification table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	ver_id	int		Yes	Yes	PK	Primary key
2	res_id	int		No	Yes	FK	FK → restaurant.res_id
3	ver_license	nvarchar	100	No	Yes		
4	ver_license	nvarchar	50	No	Yes		
5	ver_license	nvarchar	300	No	Yes		
6	ver_status	int		No	Yes		
7	ver_date_submitted	datetime		No	Yes		
8	ver_date_verified	datetime		No	Yes		
9	ver_file_attachment	nvarchar	500	No	No		
10	ver_is_deleted	bit		No	Yes		
11	ver_date_responded	datetime2(7)			No		
12	ver_content_responded	nvarchar	500		No		

z. system_log table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	sys_log_id	int		Yes	Yes	PK	Primary key
2	ad_id	int		No	No	FK	FK → admin.ad_id

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
3	action	nvarchar	100	No	No		Nullable
4	description	nvarchar(max)		No	No		Nullable
5	log_time	datetime2(3)		No	No		Nullable
6	ip_address	nvarchar	45	No	No		Nullable
7	device_info	nvarchar	255	No	No		Nullable
8	status_code	int		No	No		Nullable
9	is_success	bit		No	No		Nullable

aa. report table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	report_id	int		Yes	Yes	PK	Primary key
2	cus_id	int		No	Yes	FK	FK → customer.cus_id
3	reported_user_id	int		No	No		
4	report_type	nvarchar	50				
5	report_content	nvarchar(max)		No	No		
6	report_status	int		No	Yes		
7	report_created_at	datetime2(7)		No	Yes		
8	admin_id	int		No	No		
9	admin_note	nvarchar(max)		No	No		
10	report_related_url	nvarchar	255	No	No		
11	report_is_deleted	bit		No	Yes		

bb.contact table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	contact_id	int		Yes	Yes	PK	Primary key
2	contact_name	nvarchar	100	No	No		

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
3	contact_email	navarchar	150	No	No		
4	contact_subject	nvarchar	200	No	No		
5	contact_message	nvarchar(max)		No	No		
6	contact_status	int		No	Yes		
7	contact_reply_by_admin	nvarchar	100	No	No		
8	contact_reply_message	nvarchar(max)		No	No		
9	contact_created_at	datetime		No	Yes		
10	contact_replied_at	datetime		No	No		
11	contact_is_deleted	bit		No	Yes		

cc. dashboard_stats table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	id	int		Yes	Yes	PK	Primary key
2	report_date	datetime2(7)		No	Yes		
3	revenue_by_month_json	nvarchar(max)		No	No		
4	orders_by_month_json	nvarchar(max)		No	No		
5	customer_type_json	nvarchar(max)		No	No		
6	order_status_json	nvarchar(max)		No	No		
7	top_restaurant_json	nvarchar(max)		No	No		
8	service_revenue_json	nvarchar(max)		No	No		
9	customer_group_json	nvarchar(max)		No	No		
10	created_at	datetime2(7)		No	Yes		

dd. "table" table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	table_id	int		Yes	Yes	PK	Primary key
2	res_id	int		No	Yes	FK	FK → restaurant.res_id
3	table_name	nvarchar	100	No	Yes		
4	area_id	int		No	Yes	FK	
5	table_seat	int		No	Yes		
6	table_image	nvarchar	255	No	No		
7	table_position_x	int		No	Yes		
8	table_position_y	int		No	Yes		
9	table_is_deleted	bit		No	Yes		
10	table_created_at	datetime2(7)		No	Yes		
11	table_update_at	datetime2(7)		No	No		
12	table_status	int		No	Yes		

ee. table_area table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	area_id	int		Yes	Yes	PK	Primary key
2	res_id	int		No	Yes	FK	FK → restaurant.res_id
3	area_name	nvarchar	100	No	Yes		
4	area_description	nvarchar	255	No	No		
5	is_deleted	bit		No	Yes		
6	created_at	datetime2(7)		No	Yes		
7	updated_at	datetime2(7)		No	No		

ff. friend table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	friend_id	int		Yes	Yes	PK	Primary key
2	customer_id	int		No	Yes	FK	FK → customer.cus_id
3	friend_customer_id	int		No	Yes		
4	created_at	datetime2(7)		No	Yes		

gg.chat_message table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	chat_id	int		Yes	Yes	PK	Primary key
2	sender_id	int		No	Yes		
3	receiver_id	int		No	Yes		
4	message	varchar		No	Yes		
5	sent_at	datetime2(7)		No	Yes		
6	is_read	bit		No	Yes		

hh.priorities table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	priority_id	int			Yes	PK	Primary key
2	cus_id	int			Yes	FK	FK → customer.cus_id
3	tag	nvarchar	50		Yes		
4	count	int			Yes		
5	click_count	int			Yes		
6	last_used	datetime2(7)			No		
7	score	float			Yes		
8	weight_manual	float			No		

ii. restaurant_rating

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	rating_id	int			Yes	PK	Primary key
2	cus_id	int			Yes	FK	FK → customer.cus_id
3	res_id	int			Yes	FK	FK → restaurant.res_id
4	rating_value	int			Yes		
5	rating_comment	nvarchar(max)			Yes		
6	rating_date	datetime2(7)			No		

jj. CustomerPoint table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	point_id	int		Yes	Yes	PK	Primary key
2	cus_id	int		No	Yes	PK	FK → customer.cus_id
3	point_balance	int		No	Yes		
4	created_date	datetime		No	Yes		
5	last_updated	datetime		No	Yes		

kk. CustomerPointHistory table

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
1	history_id	int		Yes	Yes	PK	Primary key
2	point_id	int		No	Yes	PK	FK → CustomerPoint.point_id
3	change_amount	int		No	Yes		
4	balance_after_date	int		No	Yes		

#	Field name	Type	Size	Unique	Not Null	PK/FK	Notes
5	description	nvarchar	255	No	No		
6	created_date	datetime		No	Yes		

3. Detailed Design

3.1 Authorized Account

a) Class Diagram

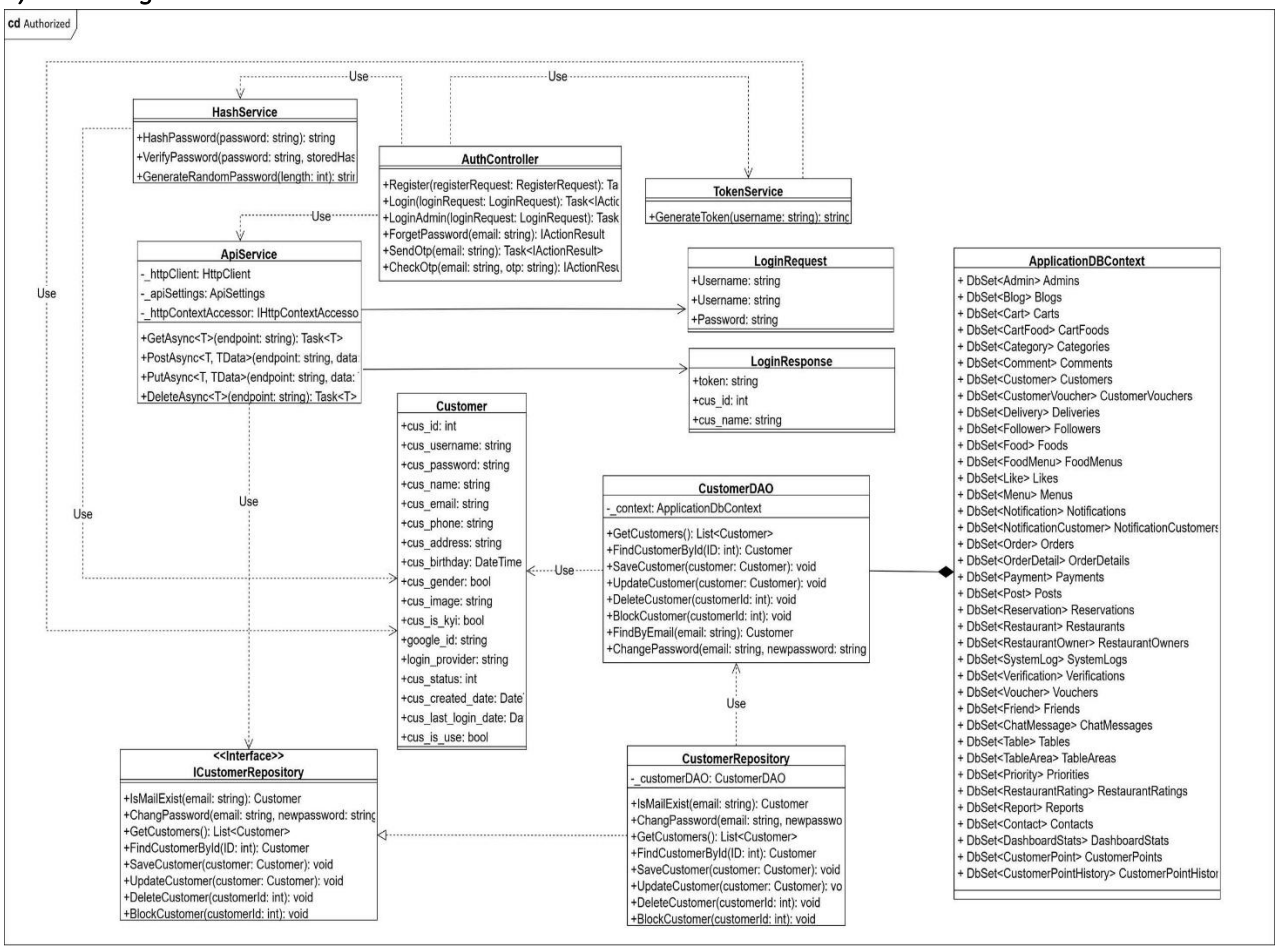


Figure 143: Class diagram authorize account

b) Sequence Diagram

Login:

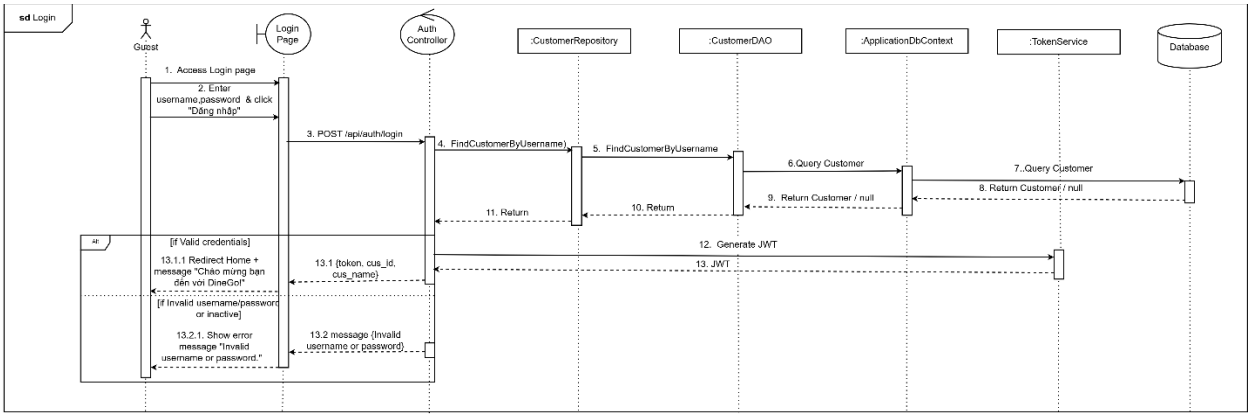


Figure 144: Sequence diagram login (Guest)

Login with google:

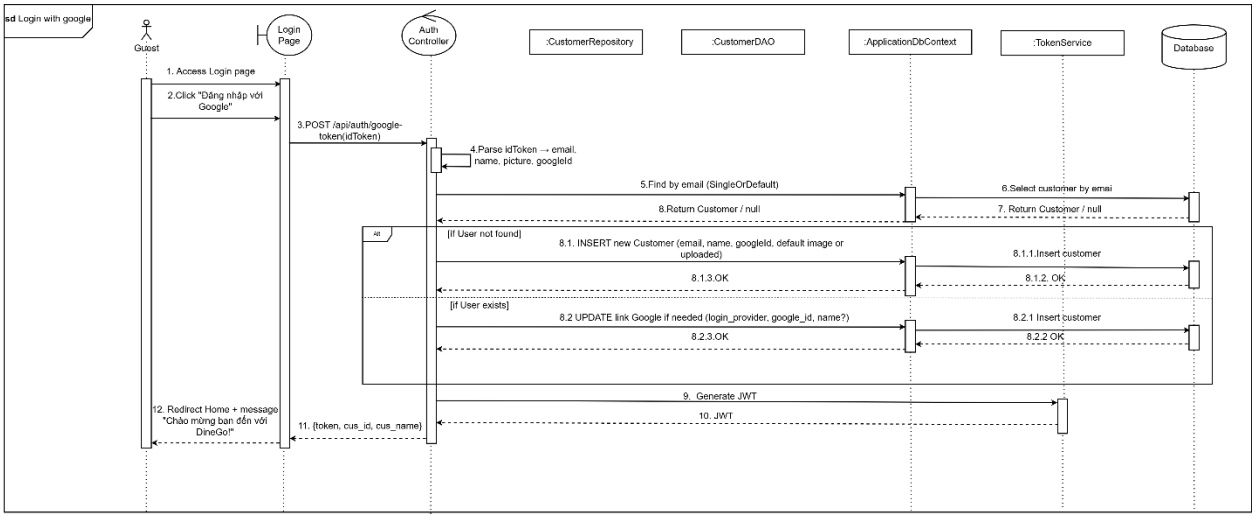


Figure 145: Sequence diagram login with google (Guest)

Register account:

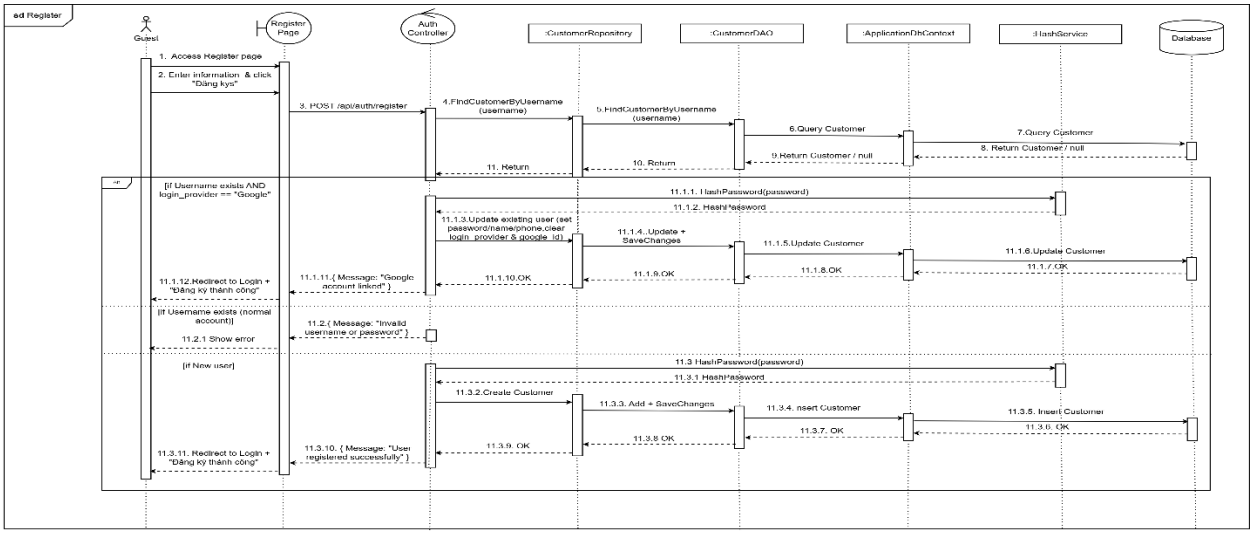


Figure 146: Sequence diagram register account (Guest)

3.2 Profile

a) Class Diagram

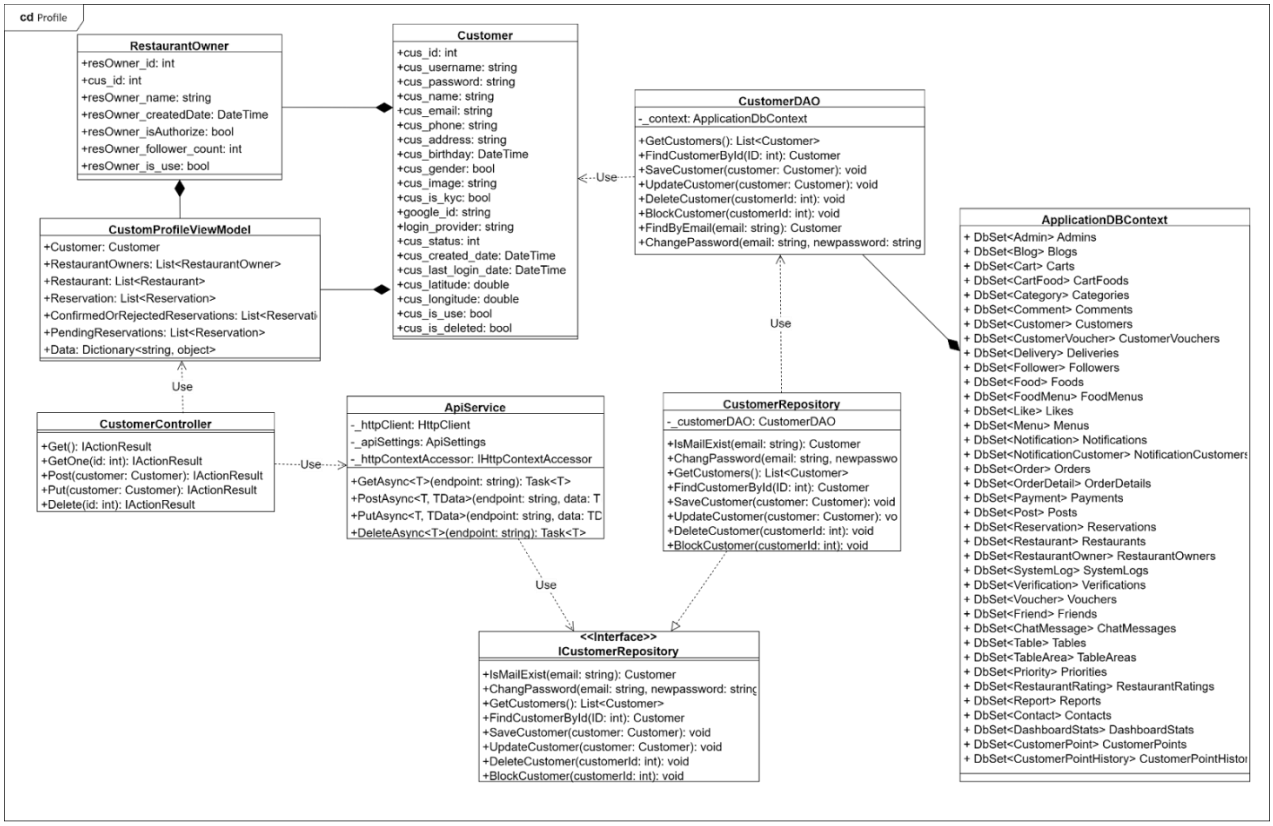


Figure 147: Class diagram profile

b) Sequence Diagram

View profile:

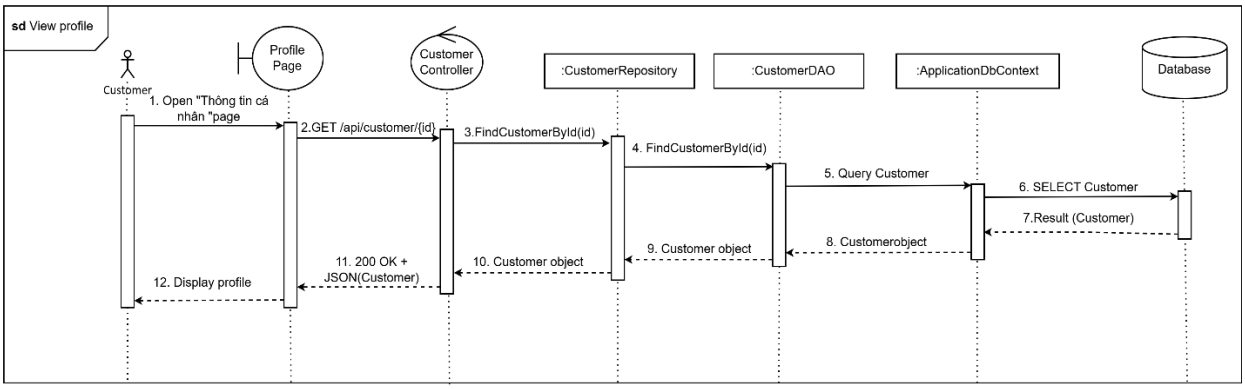


Figure 148: Sequence diagram view profile (Customer)

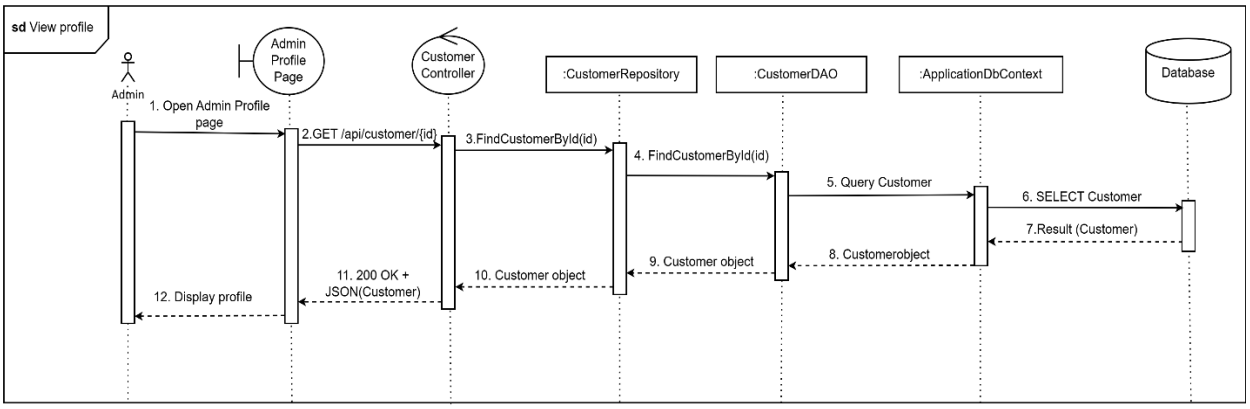


Figure 149: Sequence diagram view profile (Admin)

Modify profile:

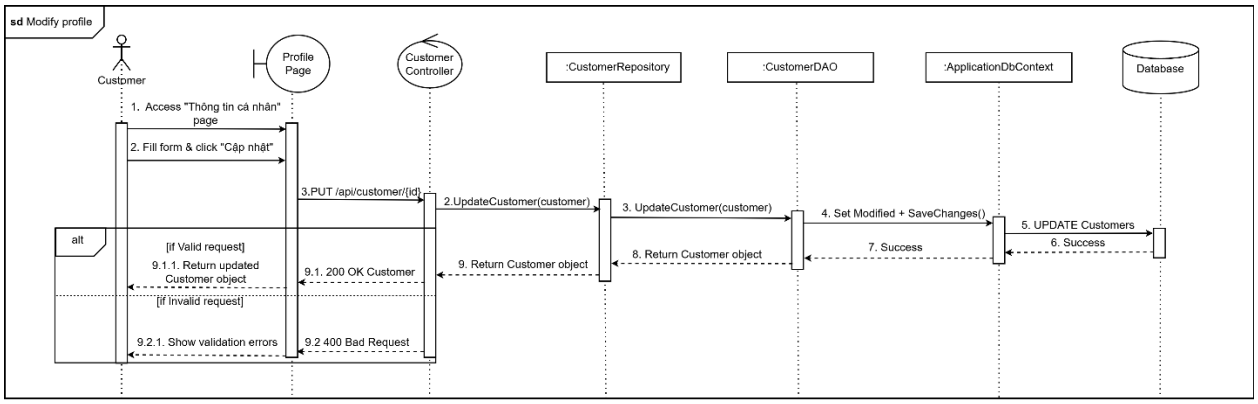


Figure 150: Sequence diagram modify profile (Customer)

3.3 Restaurant

a) Class Diagram

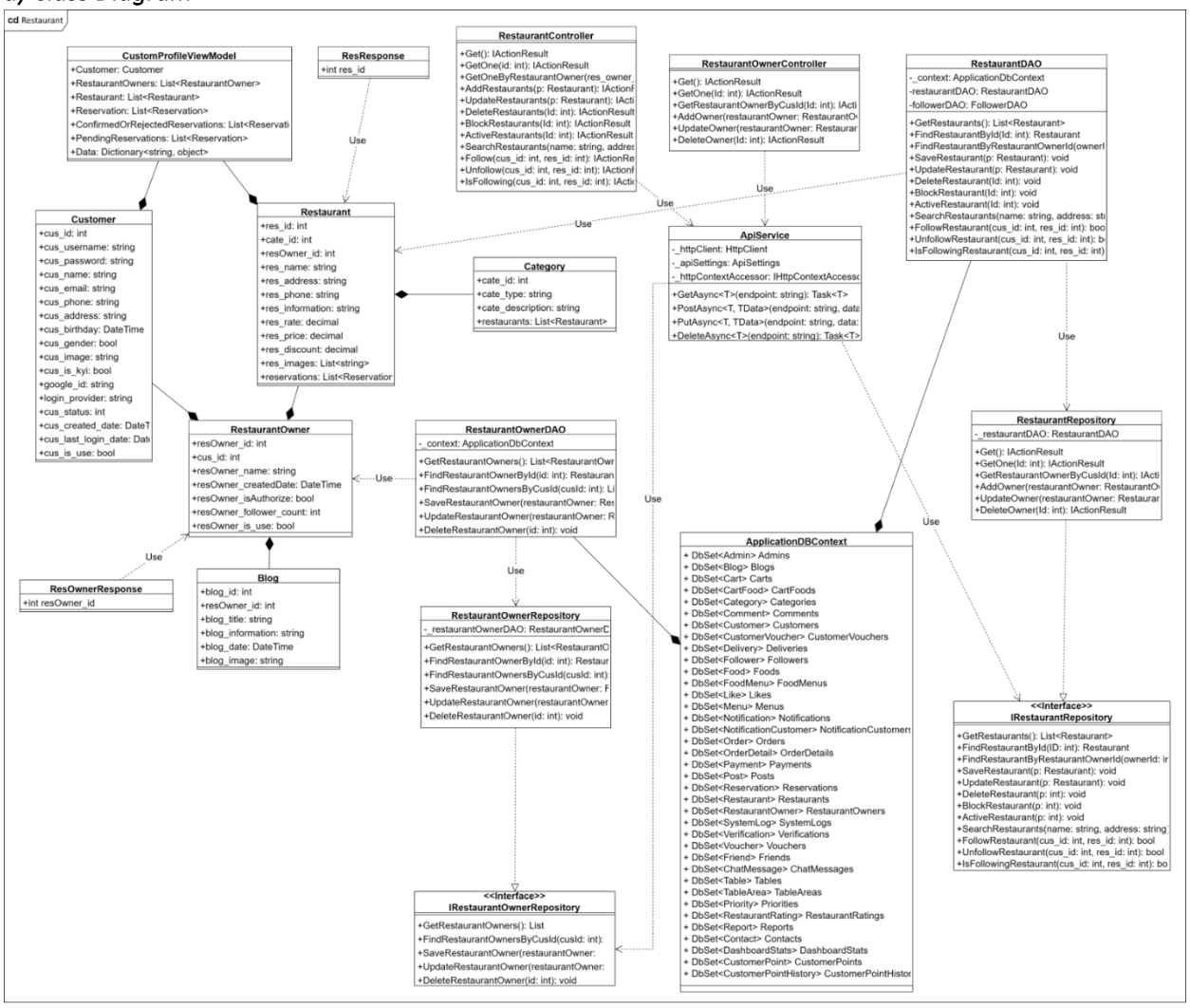


Figure 151: Class diagram of restaurant

b) Sequence Diagram

View list of restaurants:

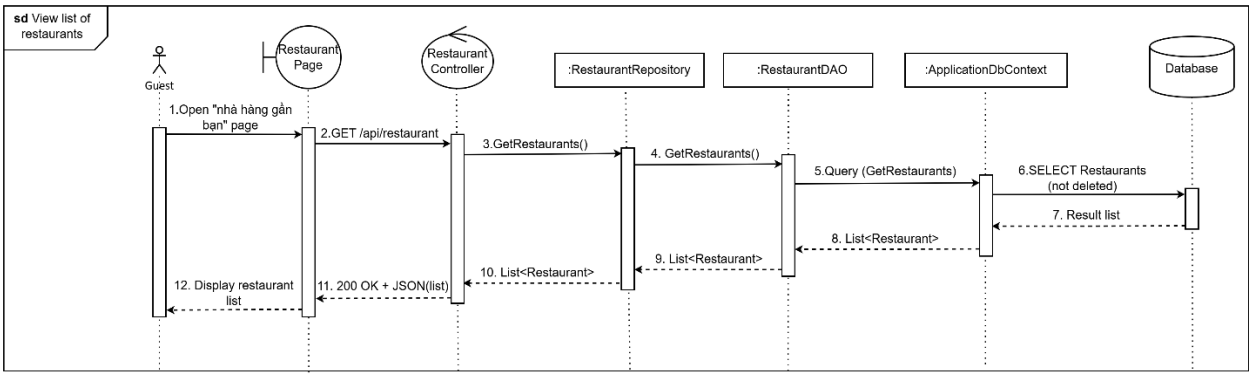


Figure 152: Sequence diagram view list of restaurants (Guest)

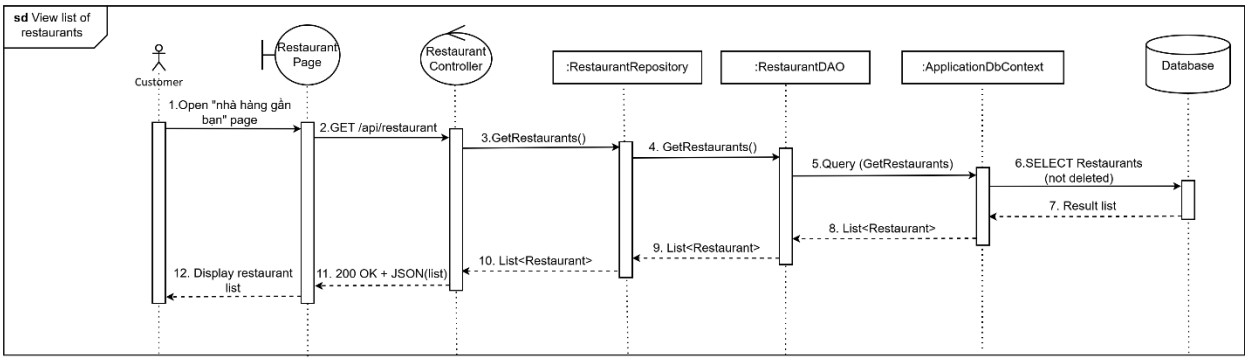


Figure 153: Sequence diagram view restaurant list of restaurants (Customer)

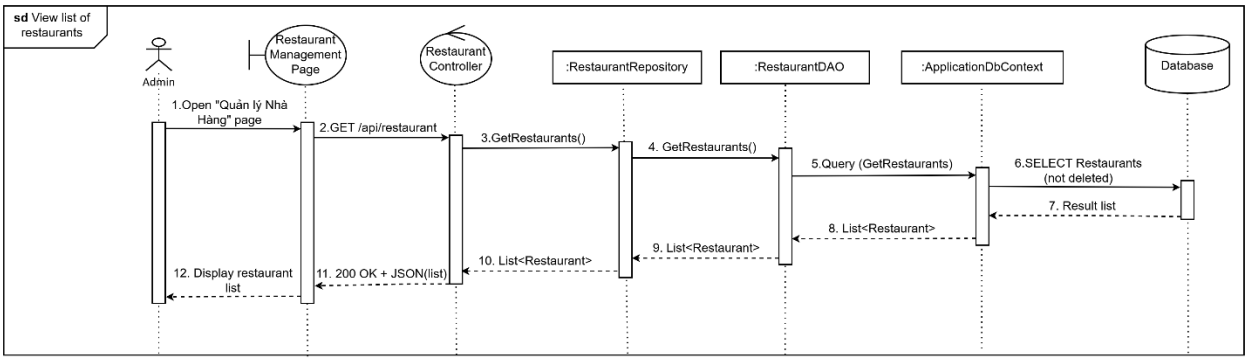


Figure 154: Sequence diagram view restaurant list of restaurants (Admin)

View restaurant detail:

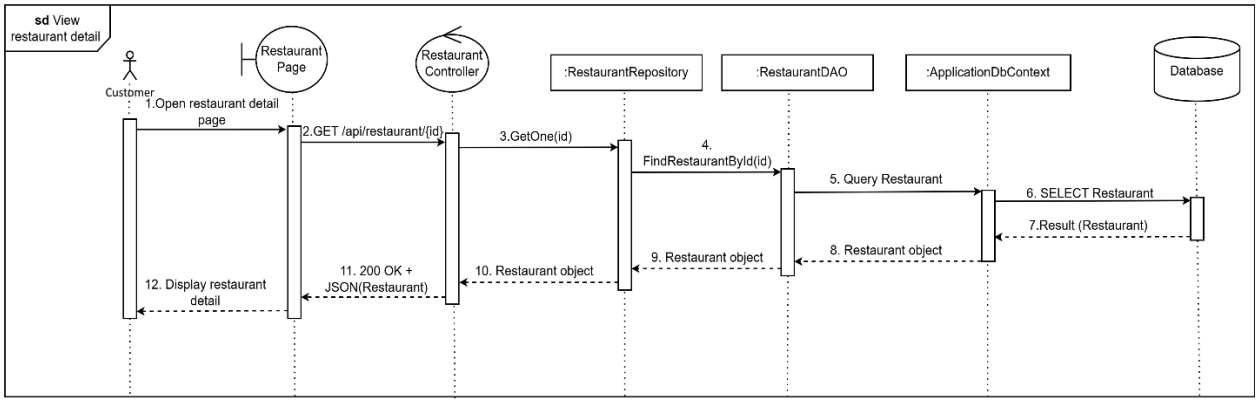


Figure 155: Sequence diagram view restaurant detail (Customer)

Search restaurant:

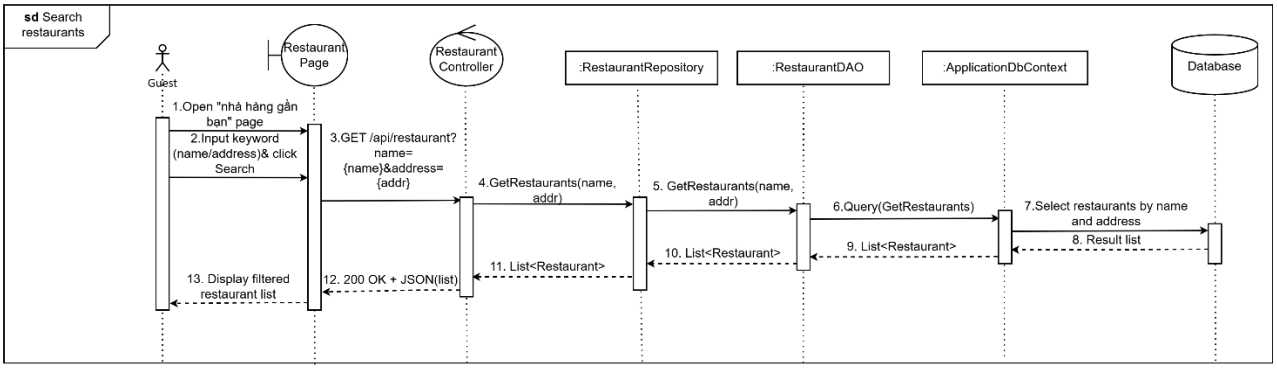


Figure 156: Sequence diagram search restaurant (Guest)

Create restaurant:

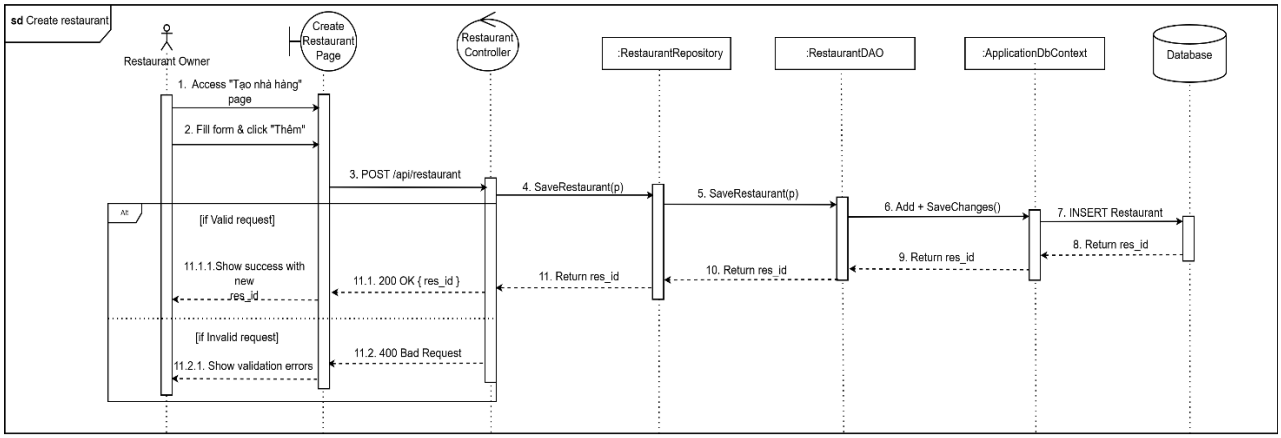


Figure 157: Sequence diagram create restaurant (Restaurant Owner)

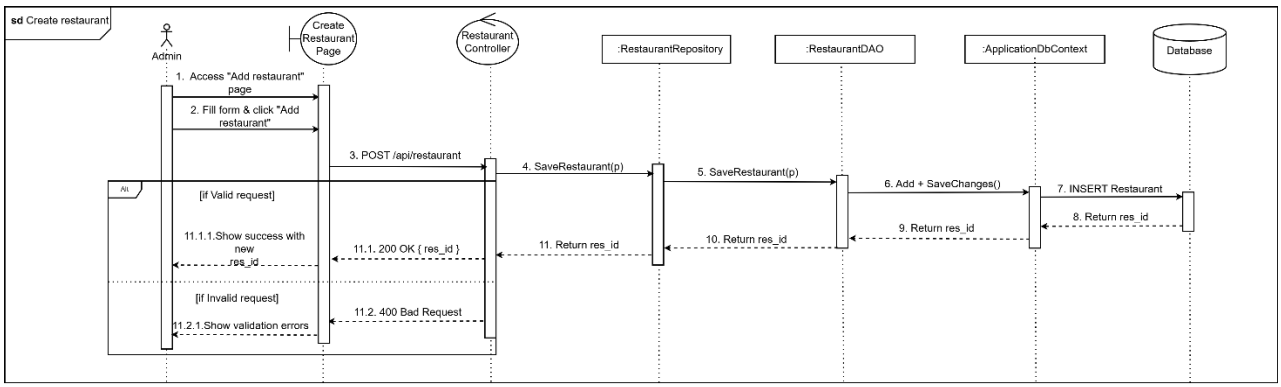


Figure 158: Sequence diagram create restaurant (Admin)

Modify restaurant:

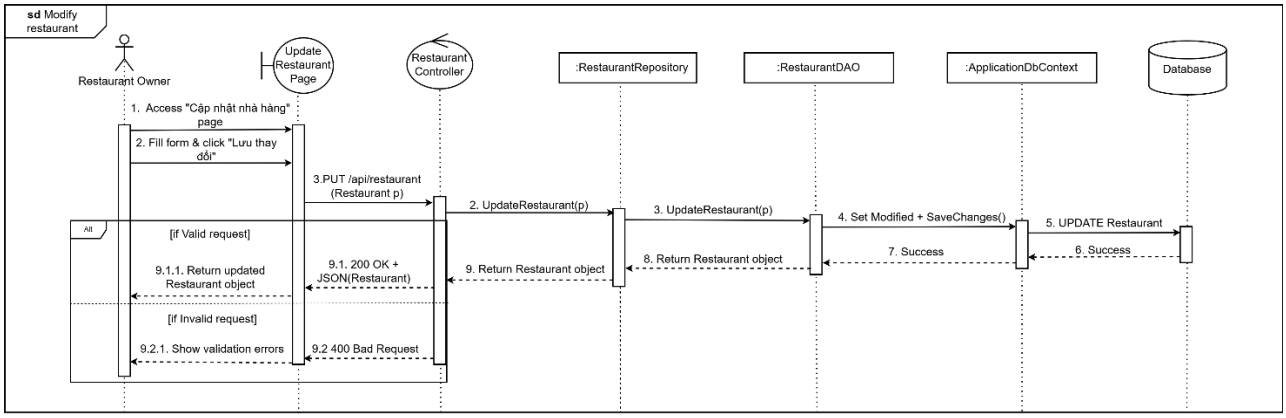


Figure 159: Sequence diagram modify restaurant (Restaurant Owner)

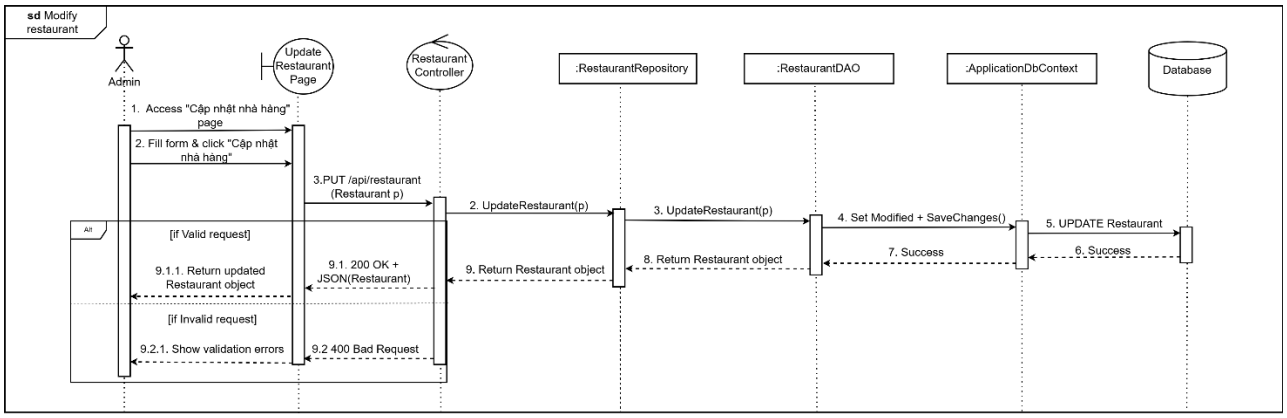


Figure 160: Sequence diagram modify restaurant (Admin)

Delete restaurant:

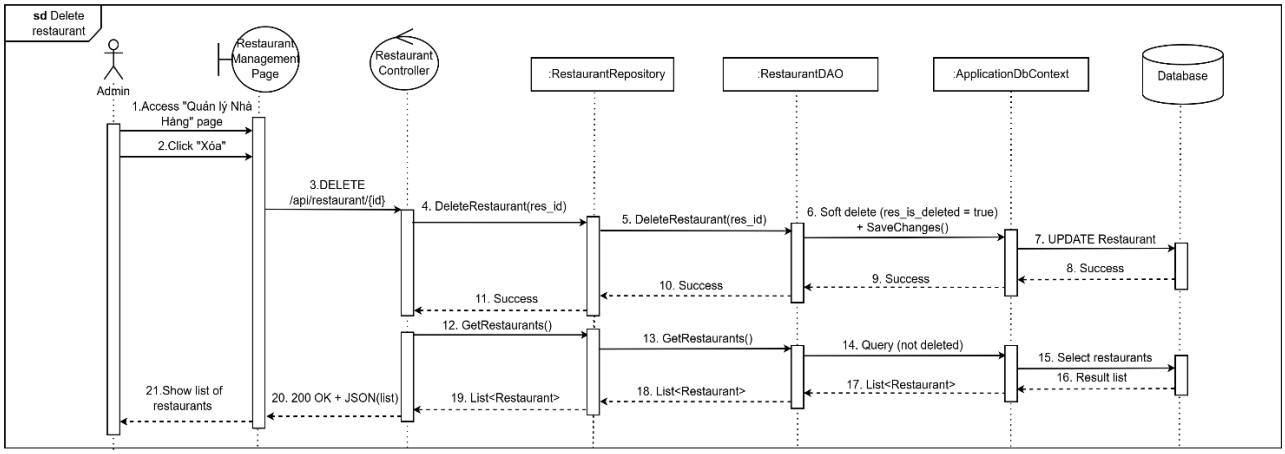


Figure 161: Sequence diagram delete restaurant (Admin)

3.4 RestaurantOwner

a) Class Diagram

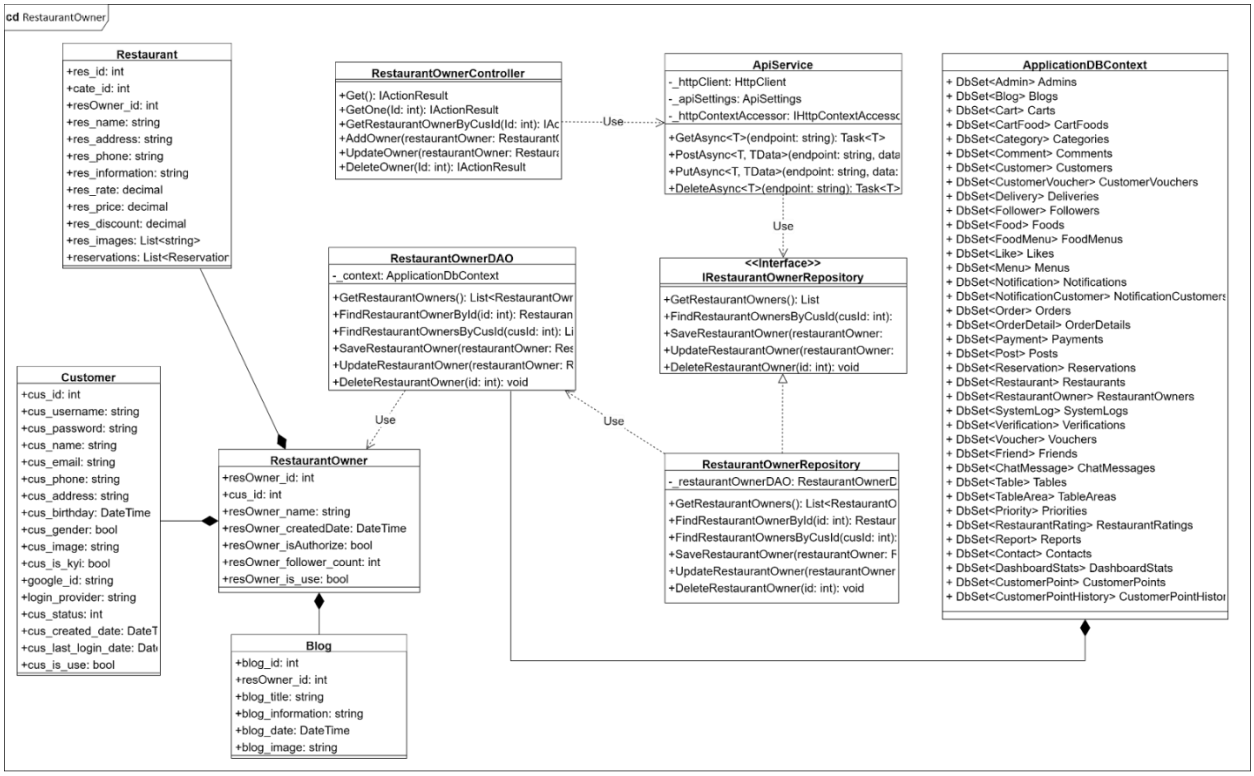


Figure 162: Class diagram of restaurant owner

b) Sequence Diagram

Register to be restaurant owner:

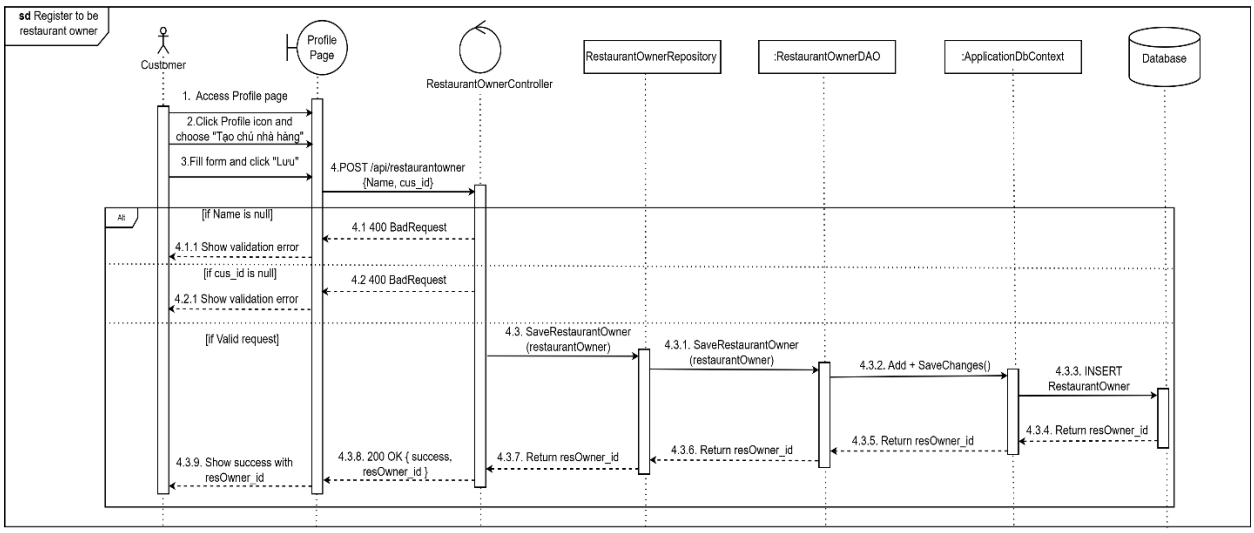


Figure 163: Sequence diagram of register to be restaurant owner (Customer)

3.5 Category

a) Class Diagram

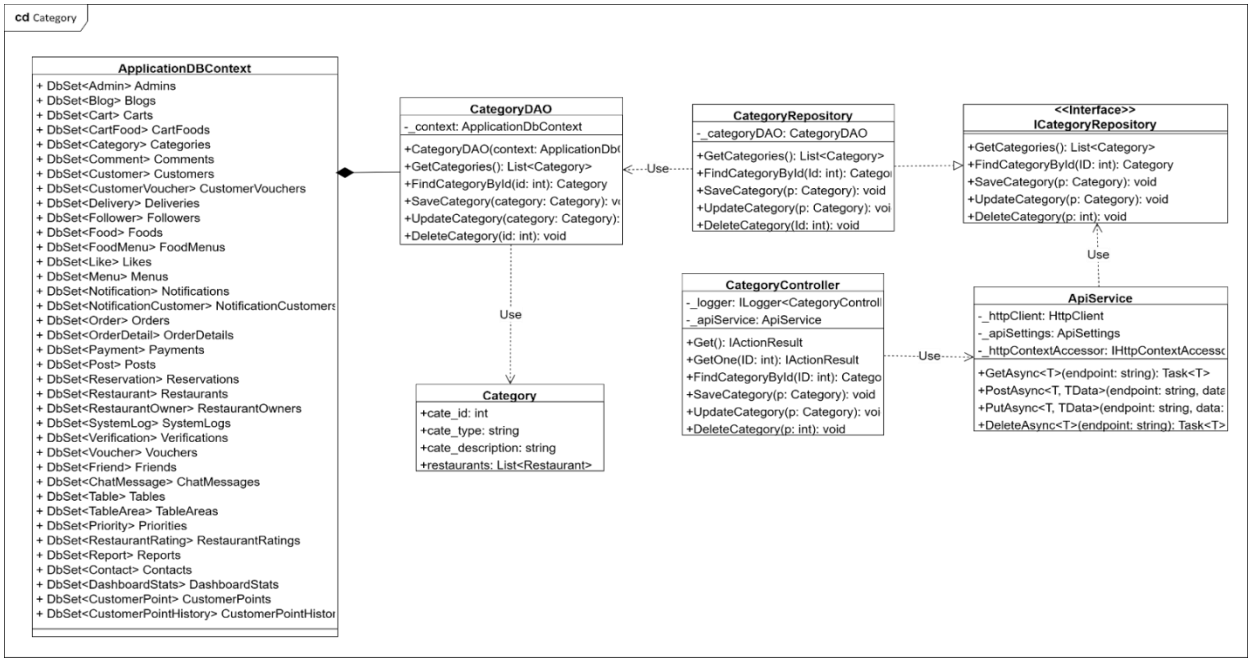


Figure 164: Class diagram of category

b) Sequence Diagram

View list of categories:

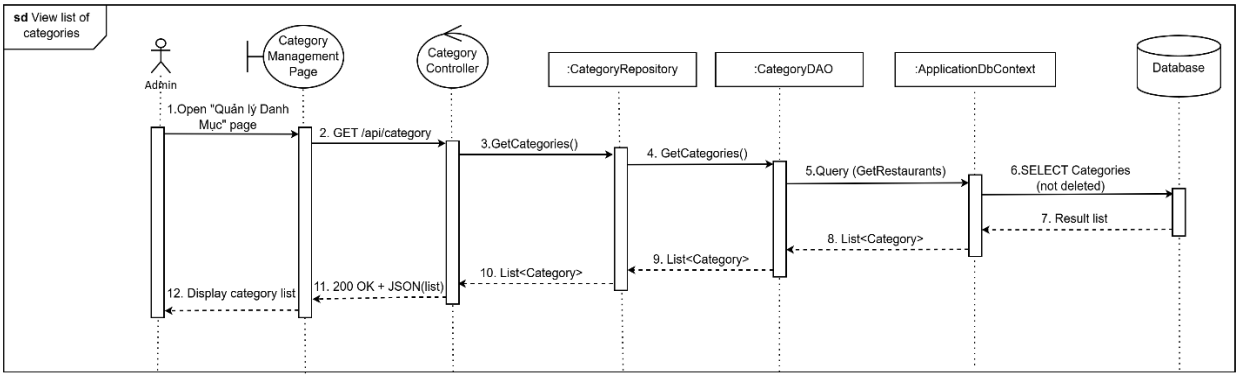


Figure 165: Sequence diagram view list of category (Admin)

Delete category:

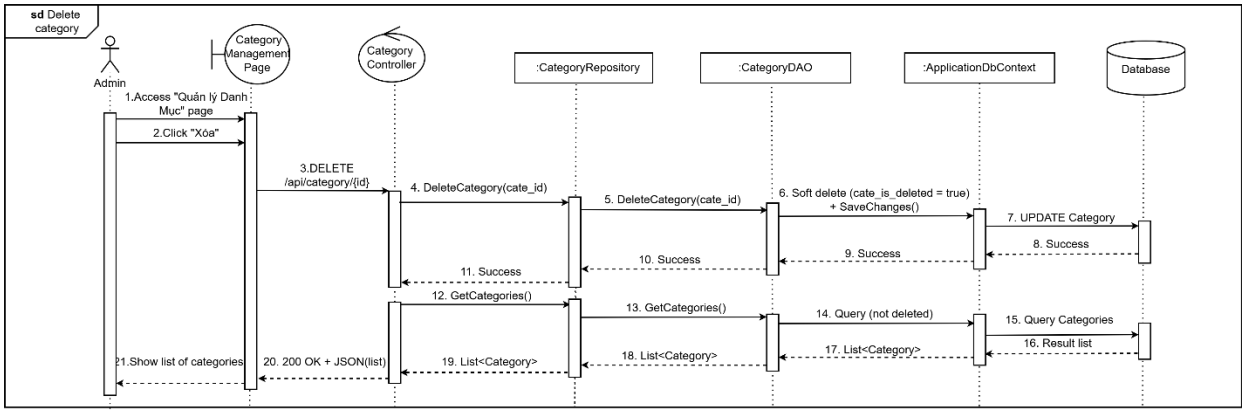


Figure 166: Sequence diagram delete category(Admin)

Create category:

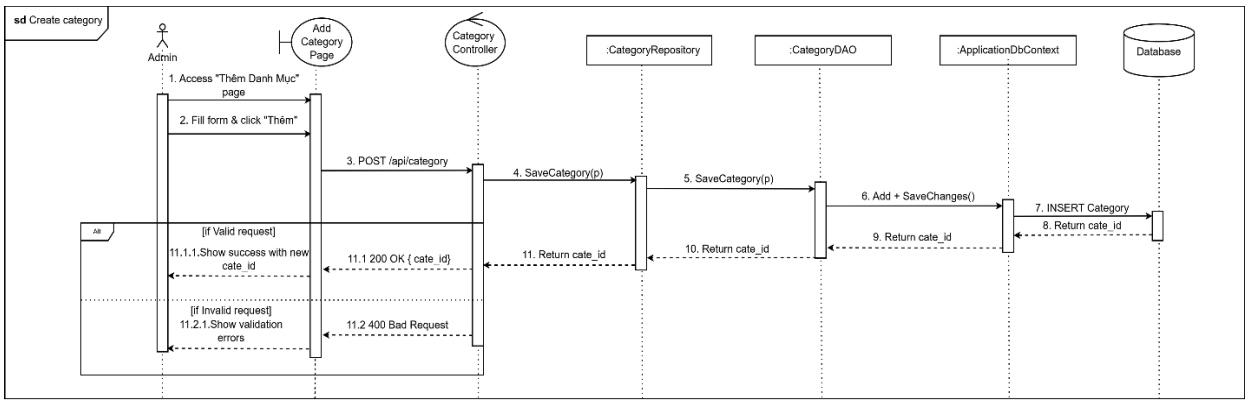


Figure 167: Sequence diagram create category (Admin)

Modify category:

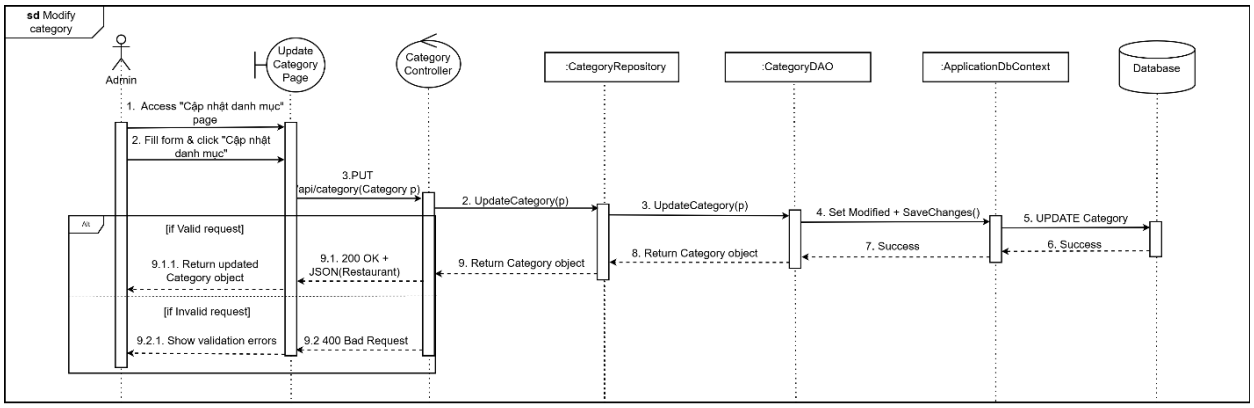


Figure 168: Sequence diagram modify category (Admin)

3.6 Blog

a) Class Diagram

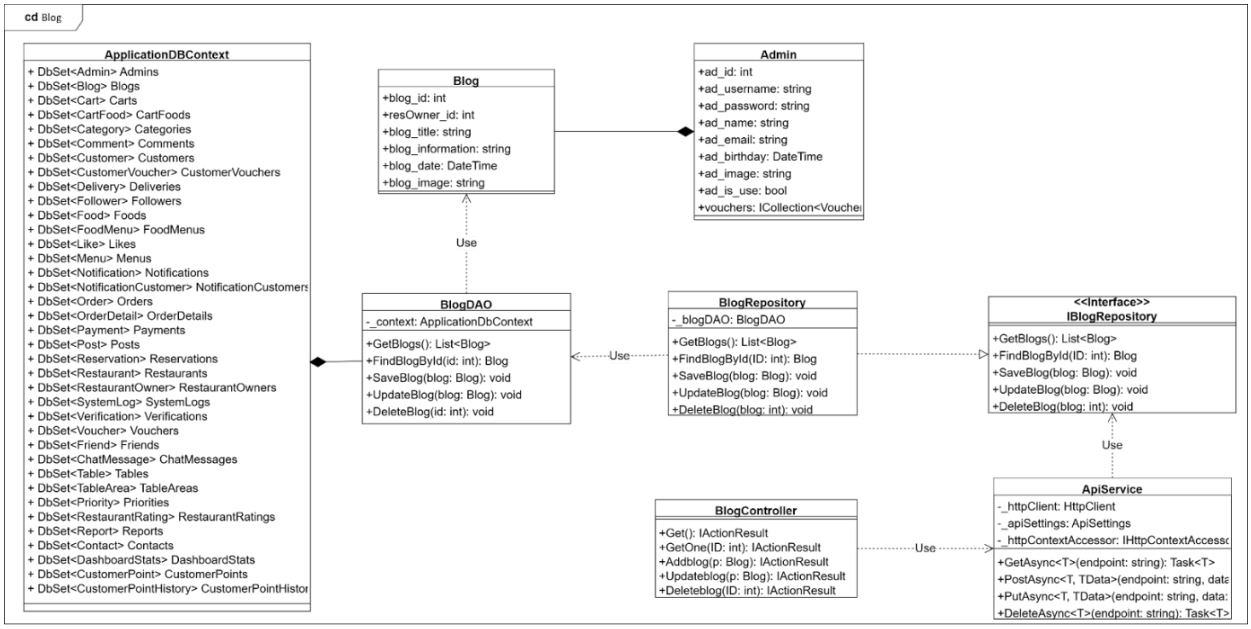


Figure 169: Class diagram of blog

b) Sequence Diagram

View list of blogs:

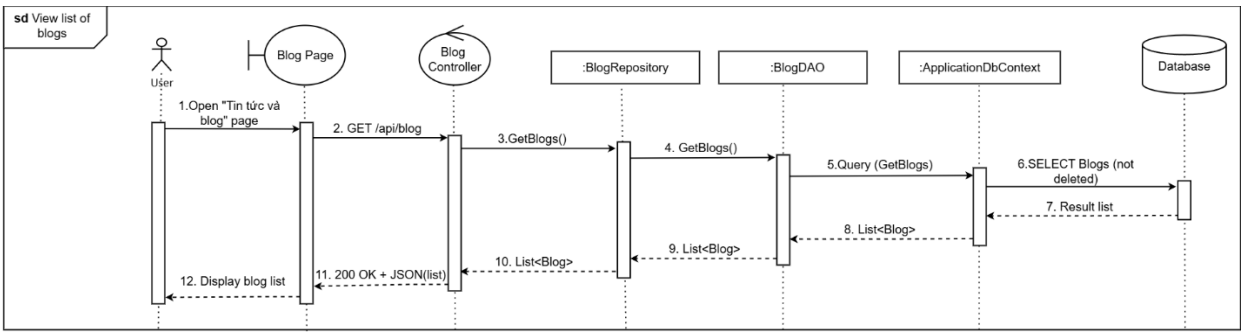


Figure 170: Sequence diagram view list of blogs (User)

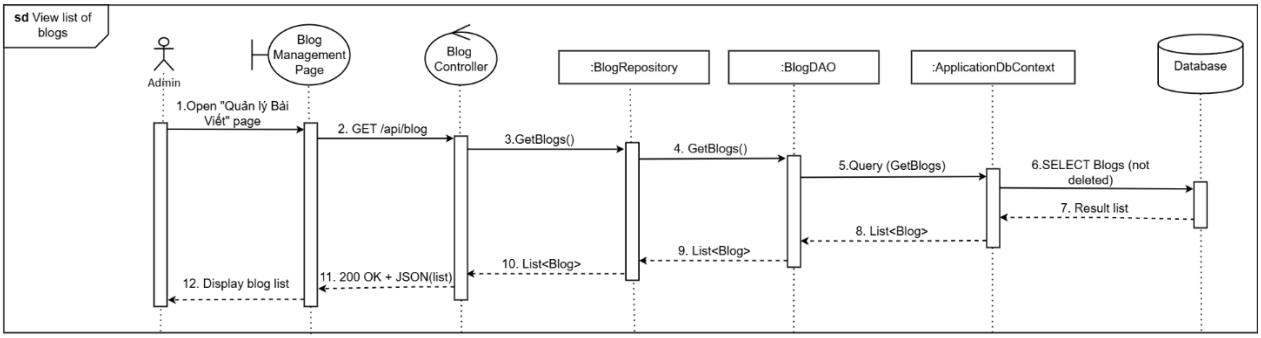


Figure 171: Sequence diagram view list of blogs (Admin)

Delete blog:

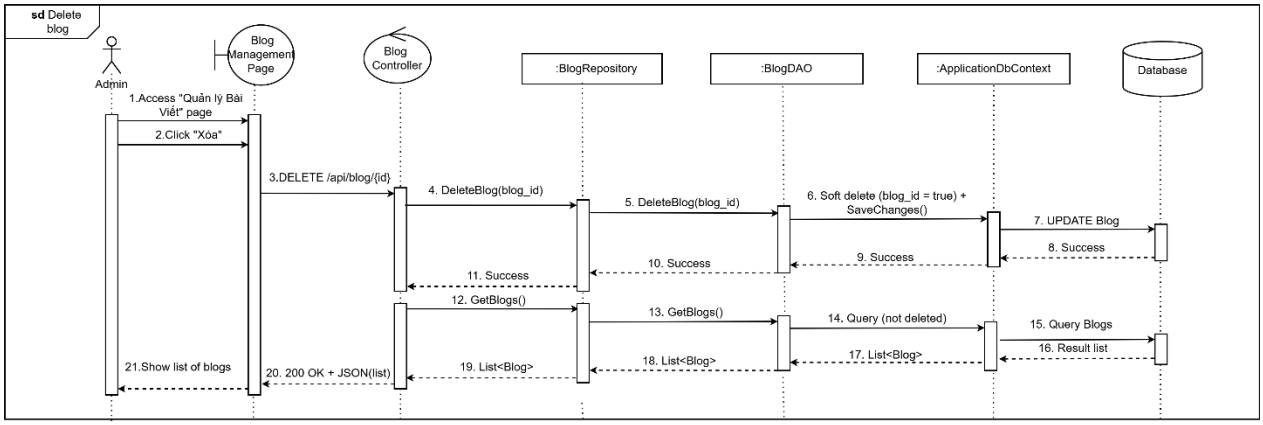


Figure 172: Sequence diagram delete blog (Restaurant Owner)

Create blog:

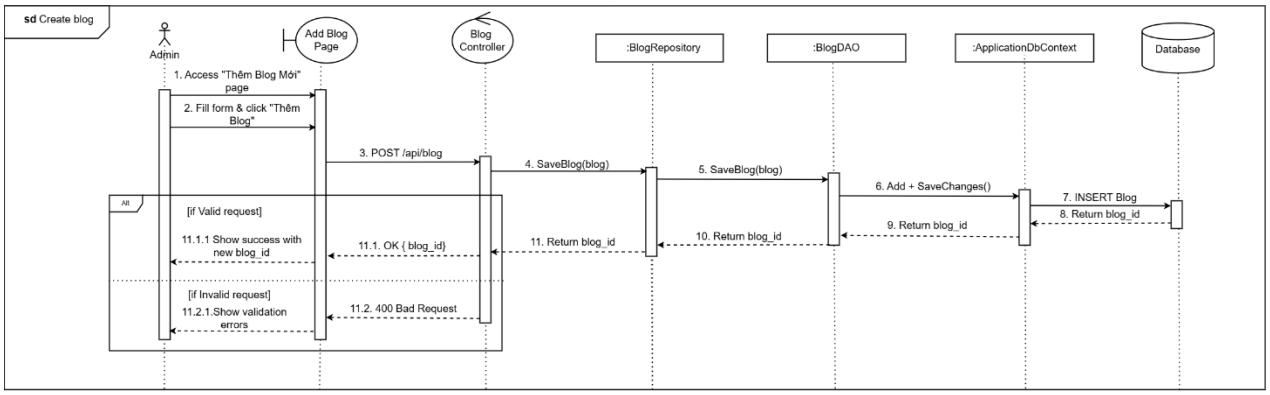


Figure 173: Sequence diagram create blog (Admin)

Modify blog:

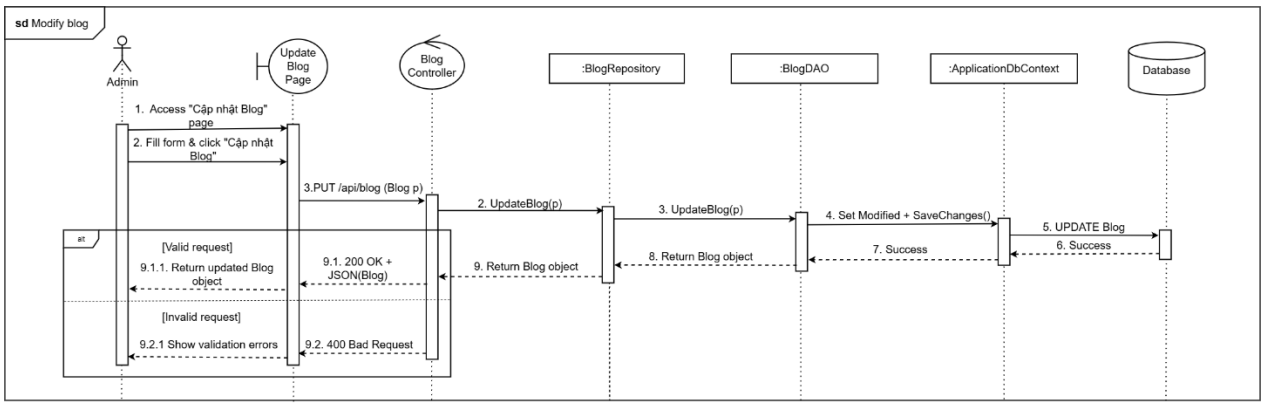


Figure 174: Sequence diagram modify blog (Admin)

3.6 Reservation

a) Class Diagram

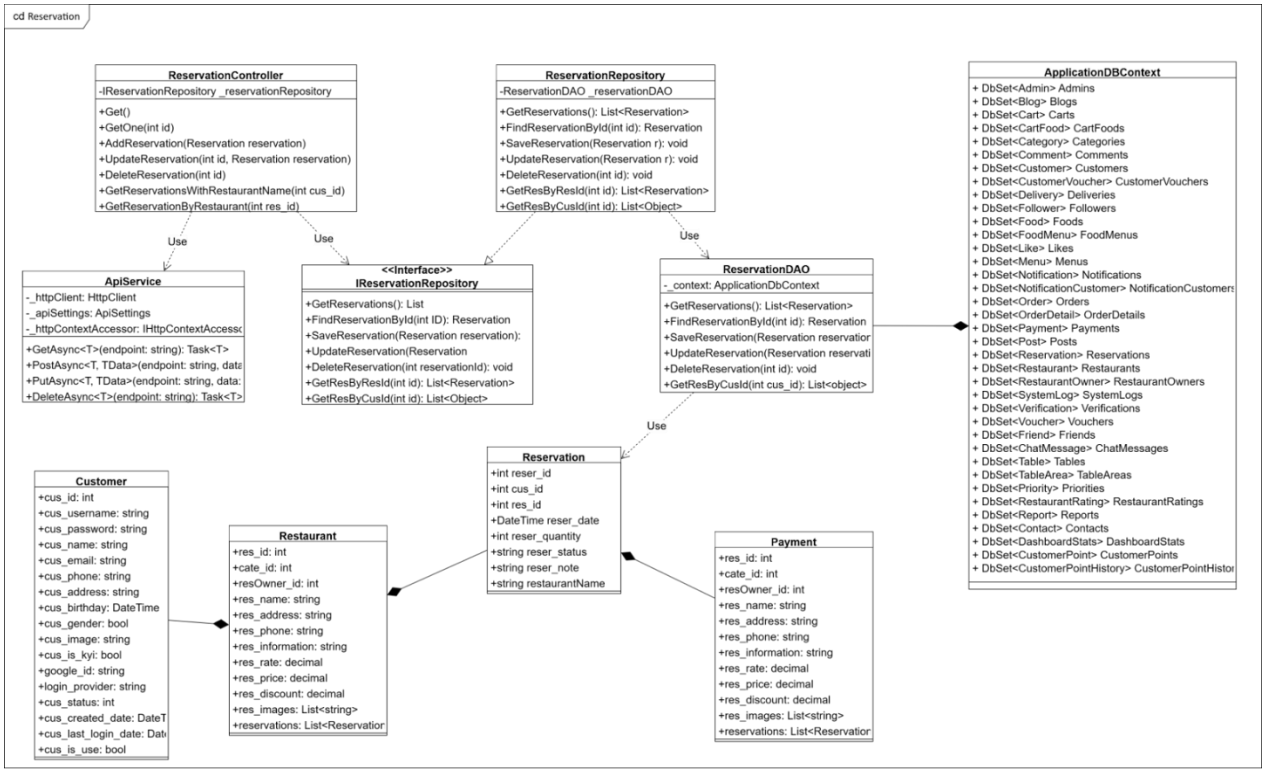


Figure 175: Class diagram of reservation

b) Sequence Diagram

View list of reservations:

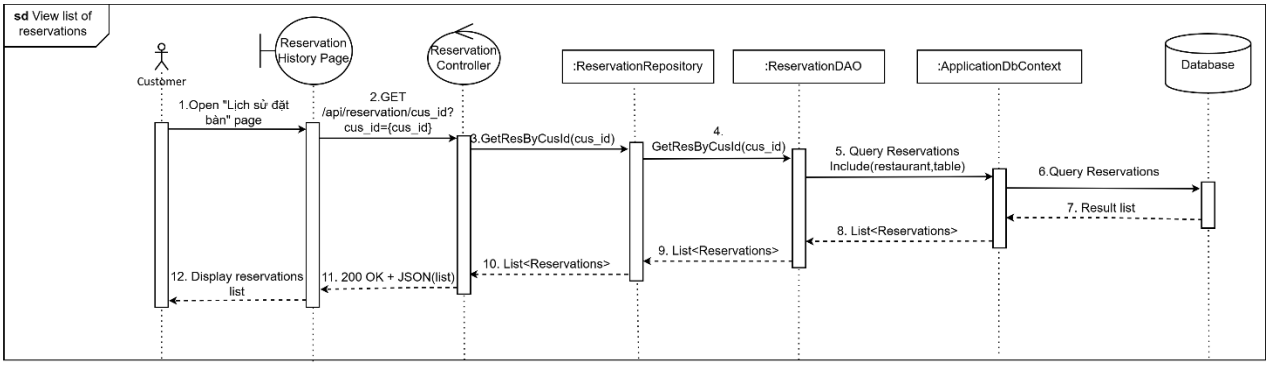


Figure 176: Sequence diagram view list of reservations (Customer)

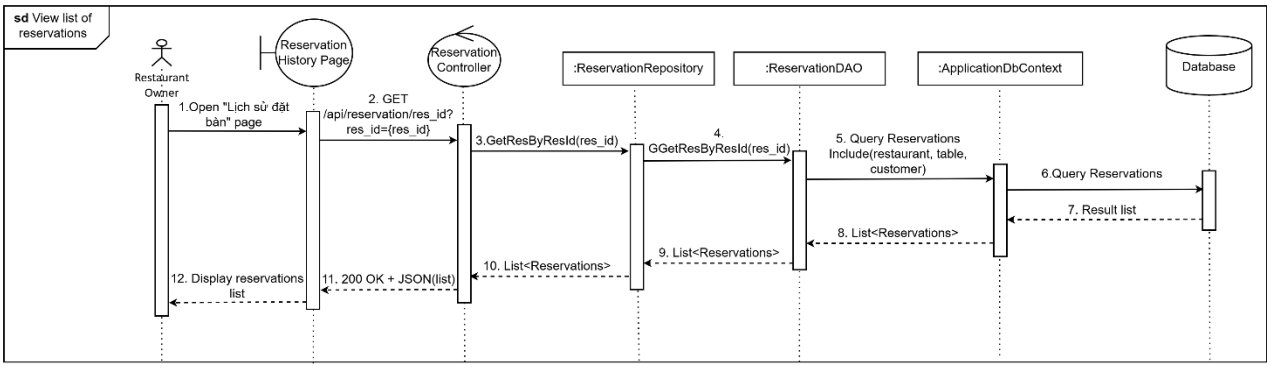


Figure 177: Sequence diagram view list of reservations (Restaurant Owner)

Create reservation:

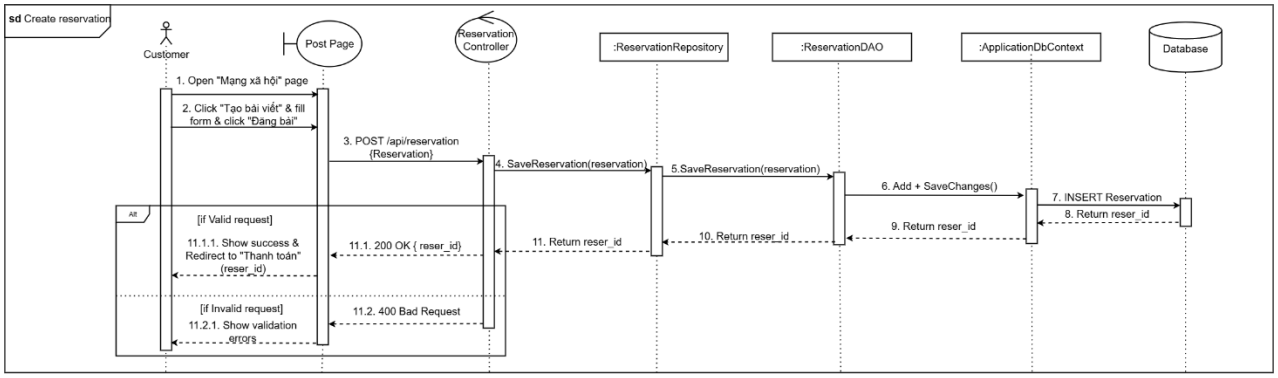


Figure 178: Sequence diagram create reservation (Customer)

Modify reservation:

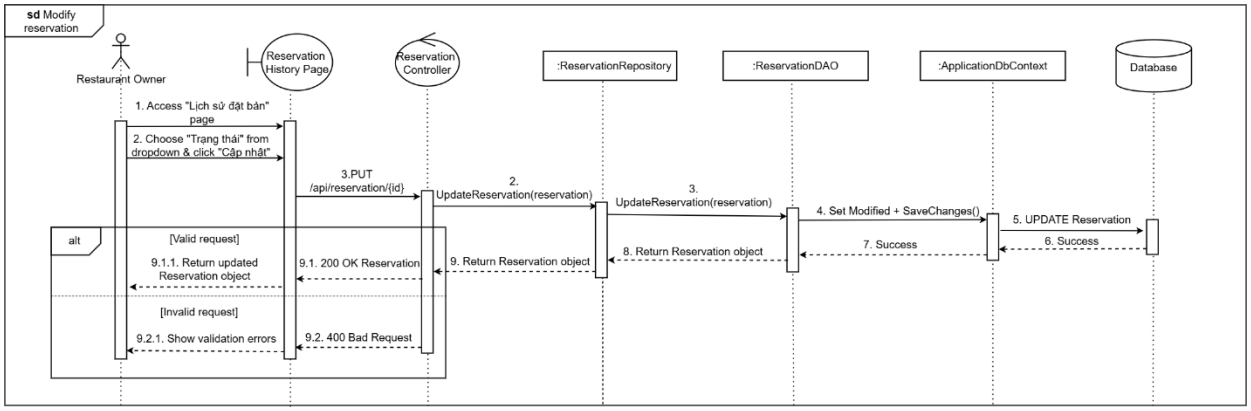


Figure 179: Sequence diagram modify reservation (Restaurant Owner)

3.7 Post

a) Class Diagram

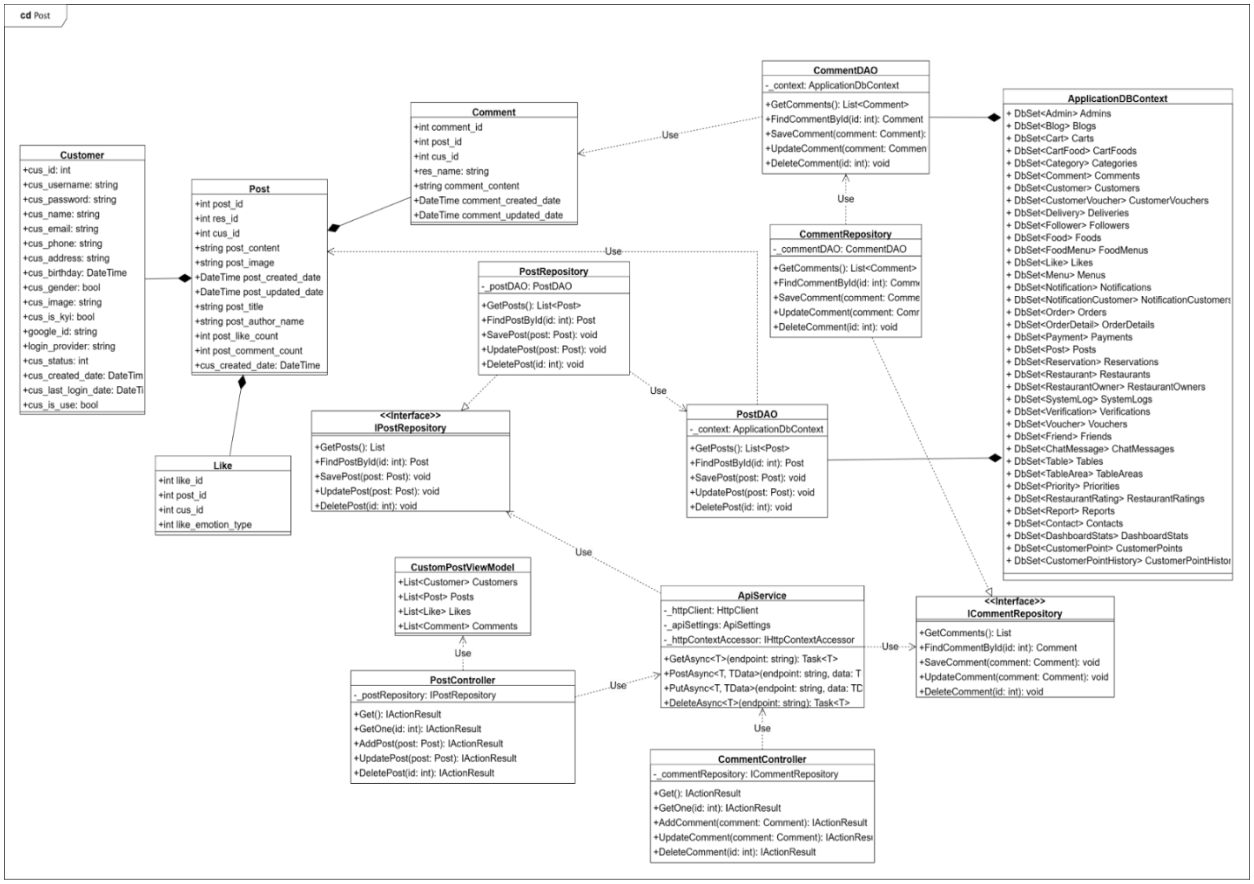


Figure 180: Class diagram of post

b) Sequence Diagram

View list of posts:

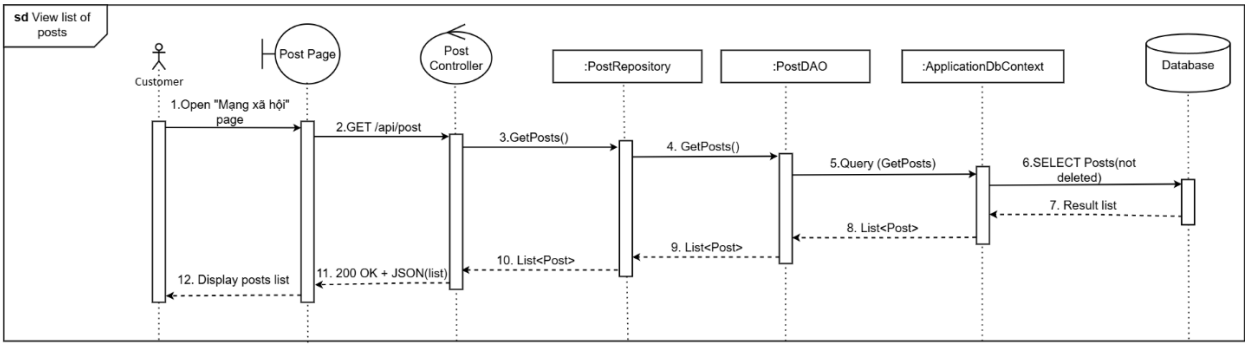


Figure 181: Sequence diagram view list of posts (Customer)

Delete post:

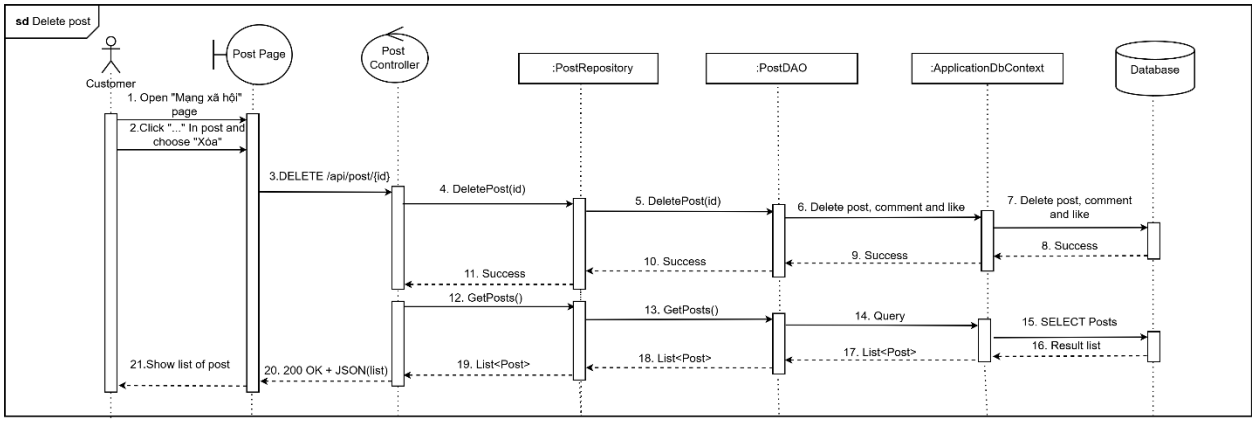


Figure 182: Sequence diagram delete post (Customer)

Create post:

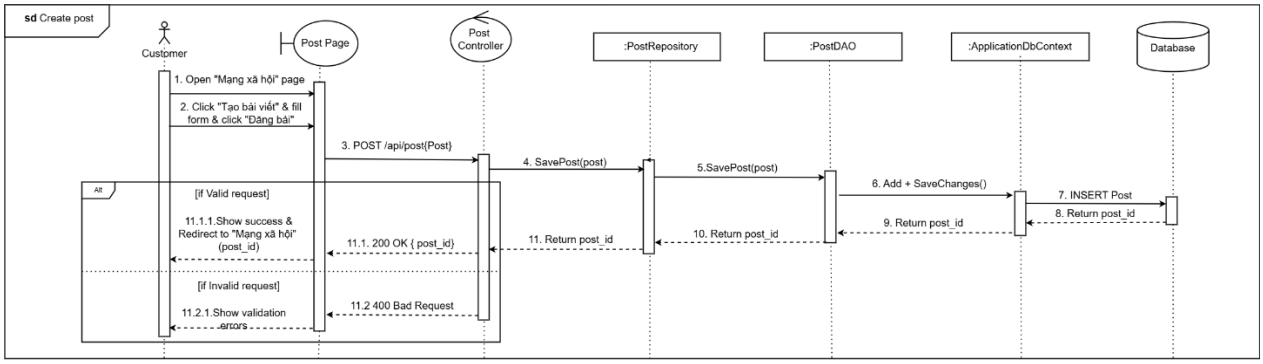


Figure 183: Sequence diagram create post (Customer)

Modify post:

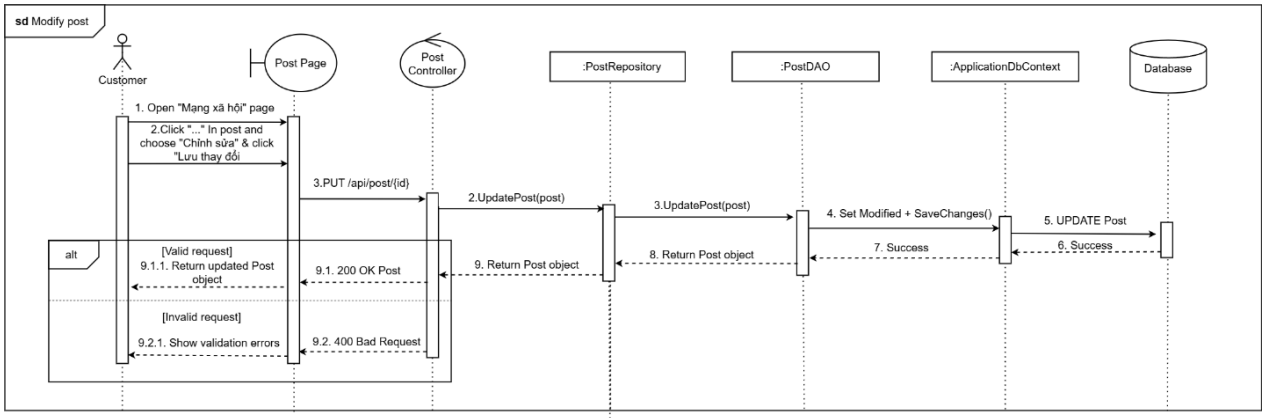


Figure 184: Sequence diagram modify post (Customer)

3.8 Food

a) Class Diagram

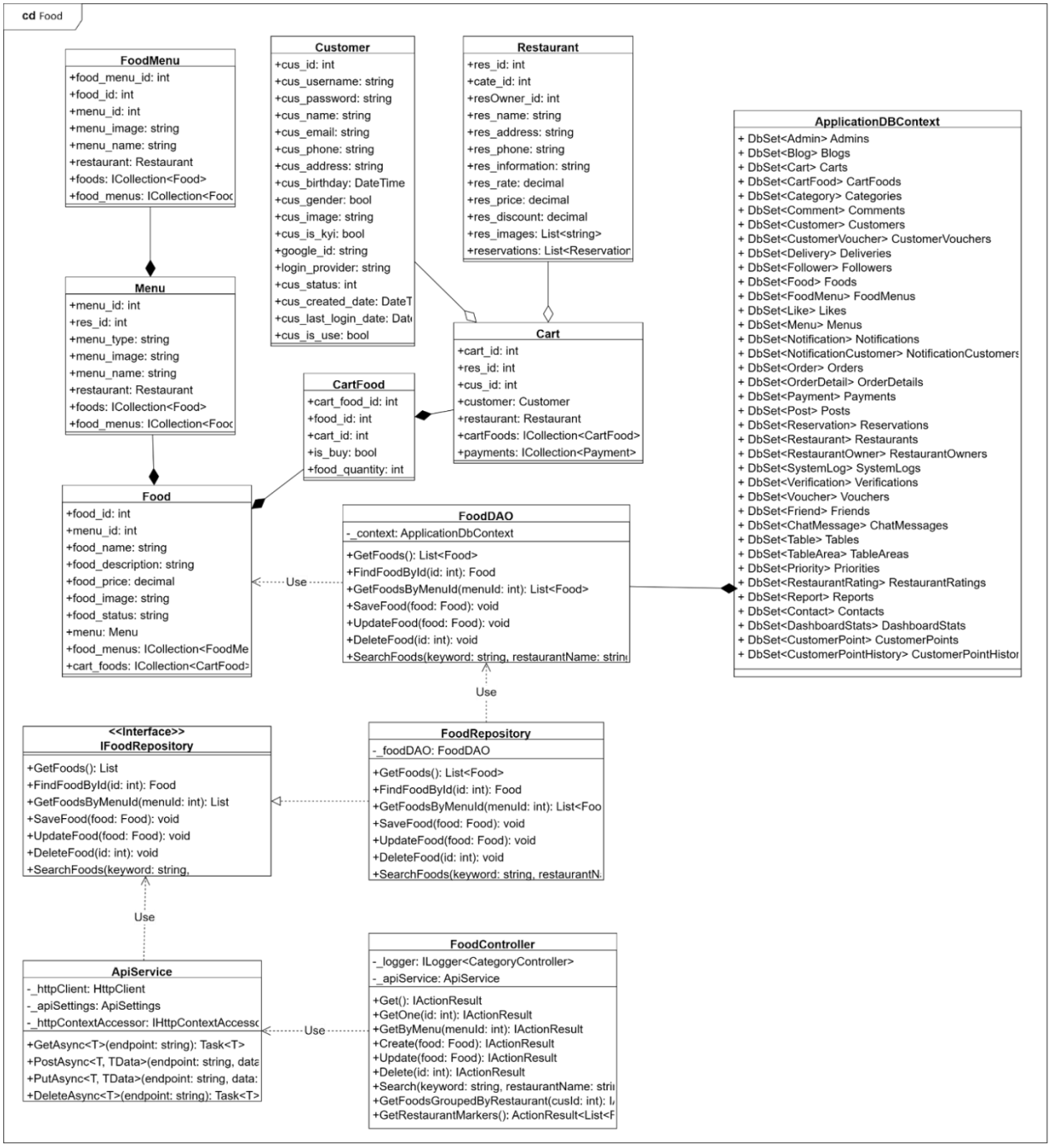


Figure 185: Class diagram of food

b) Sequence Diagram

Search food:

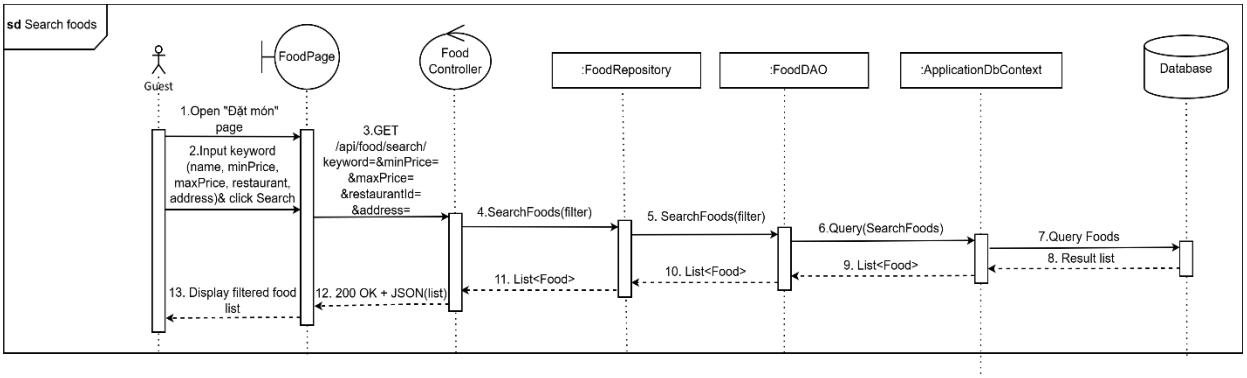


Figure 186: Sequence diagram search foods (Guest)

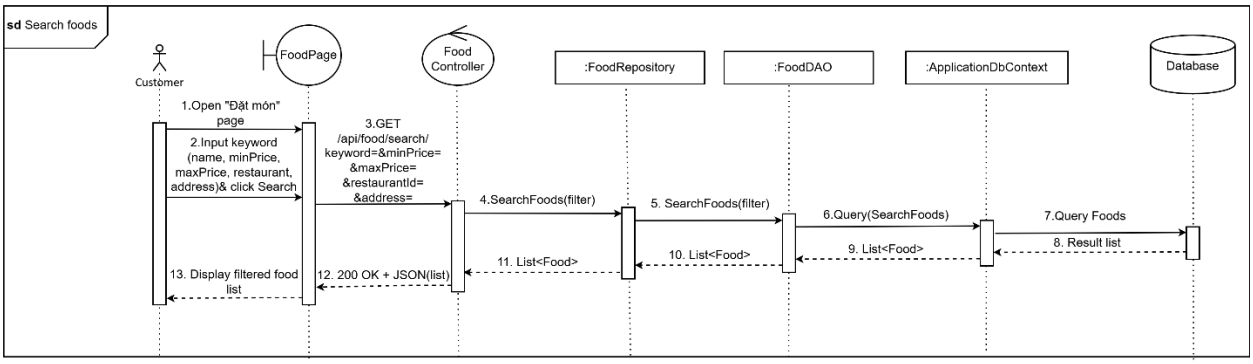


Figure 187: Sequence diagram search foods (Customer)

Search foods by distance:

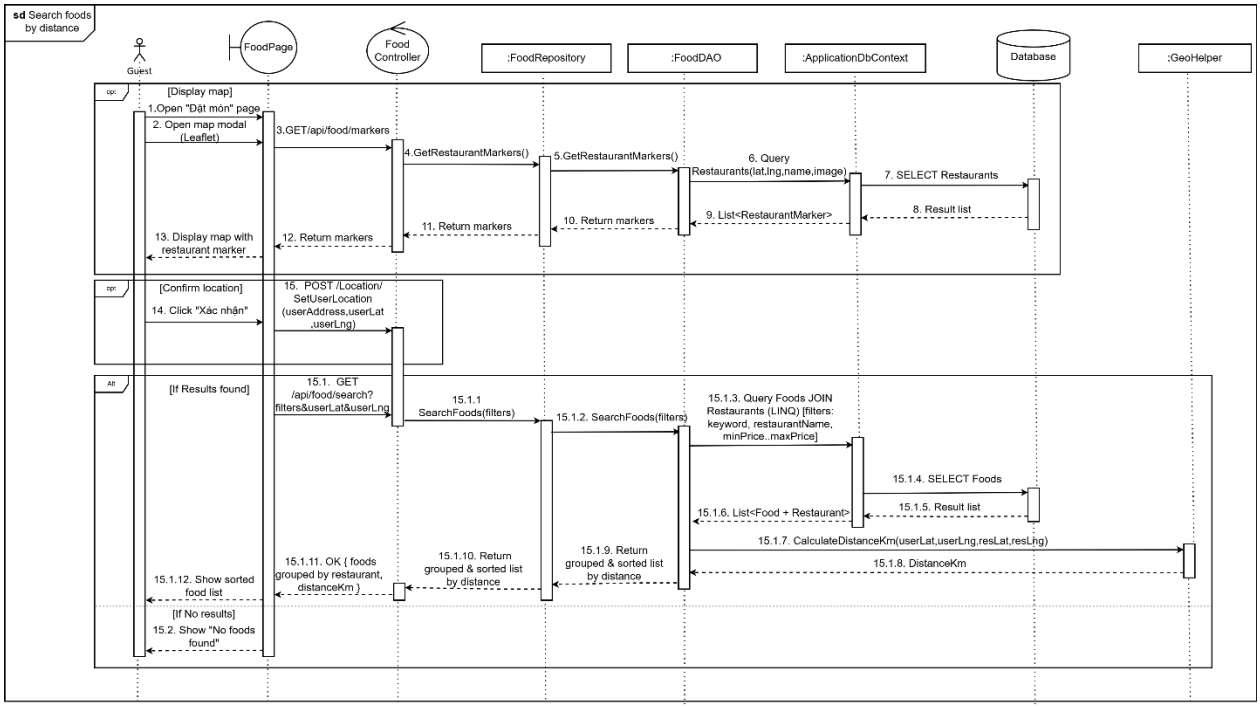


Figure 188: Sequence diagram Search foods by distance (Guest)

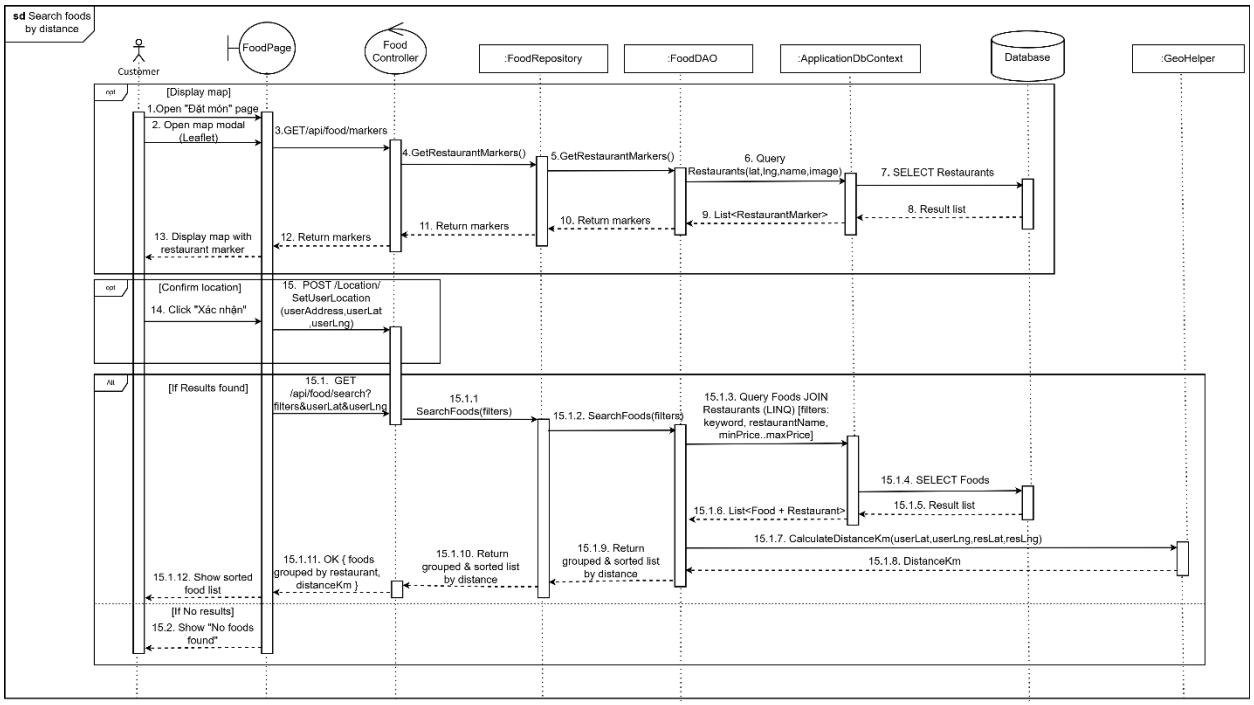


Figure 189: Sequence diagram search foods by distance (Customer)

View list of foods:

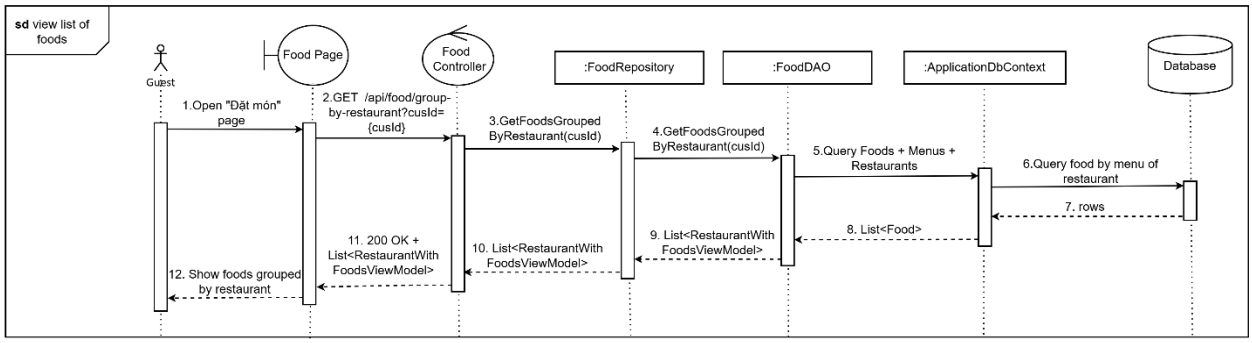


Figure 190: Sequence diagram view list of foods (Guest)

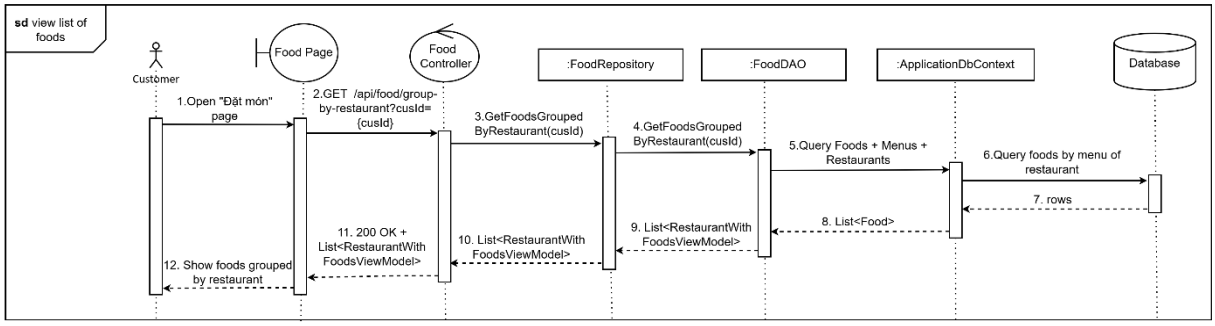


Figure 191 Sequence diagram view list of foods (Customer)

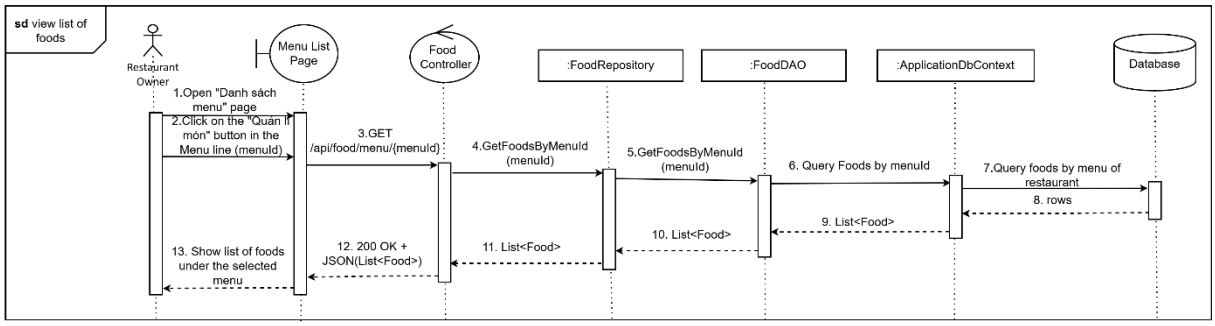


Figure 192: sequence diagram view list of foods (Restaurant Owner)

View food detail:

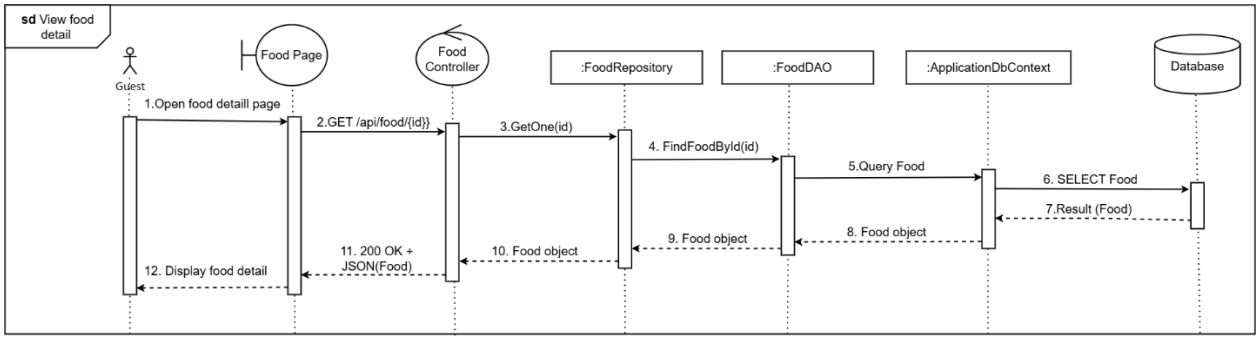


Figure 193: Sequence diagram view food detail (Guest)

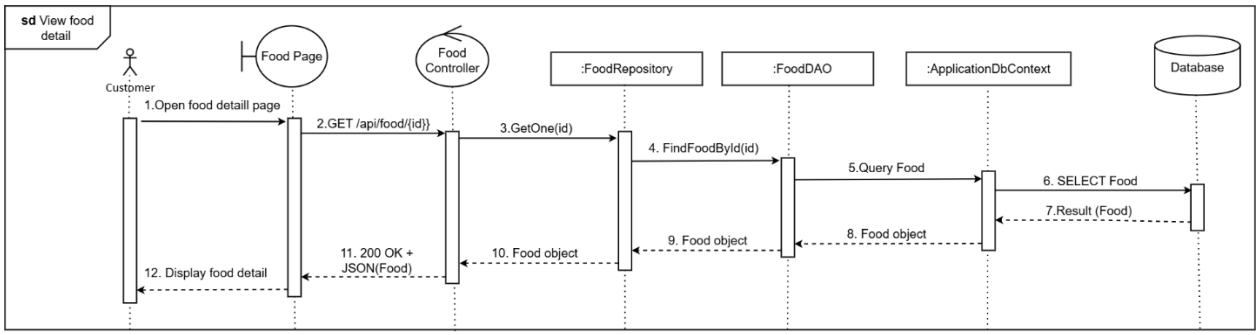


Figure 194: Sequence diagram view food detail (Customer)

Create food:

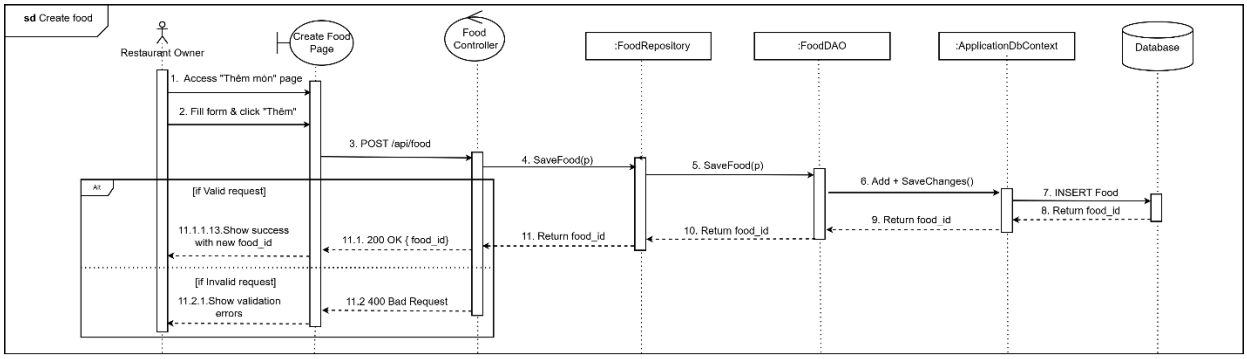


Figure 195: sequence diagram create food (Restaurant Owner)

Modify food:

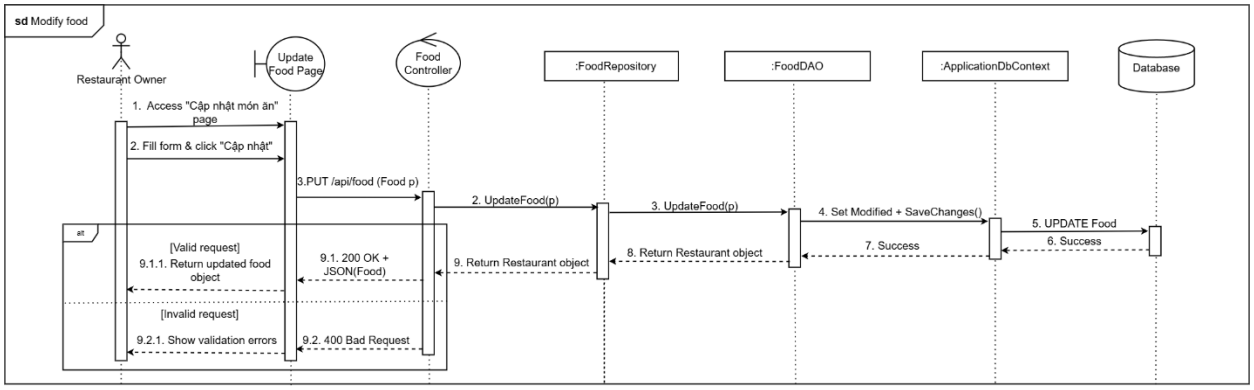


Figure 196: Sequence diagram modify food (Restaurant Owner)

Delete food:

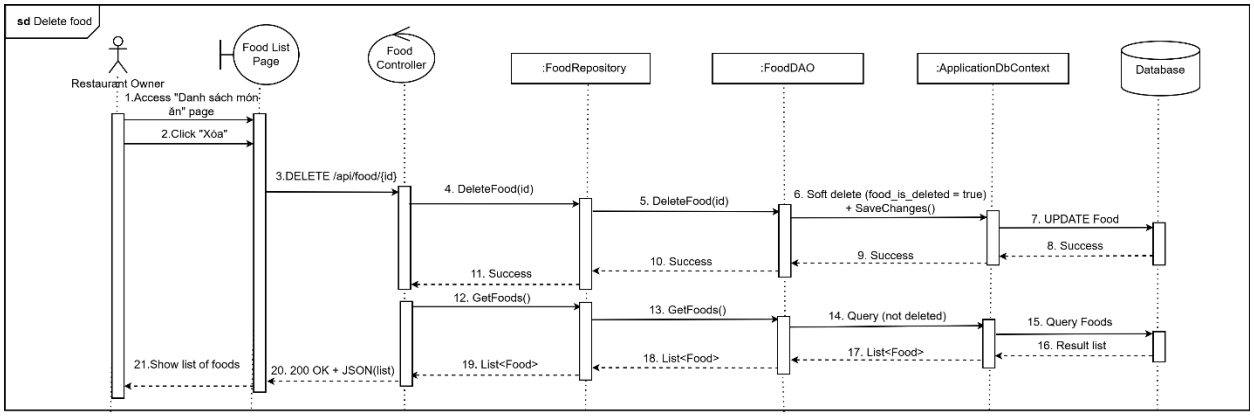


Figure 197: Sequence diagram delete food (Restaurant Owner)

3.9 Table

a) Class Diagram

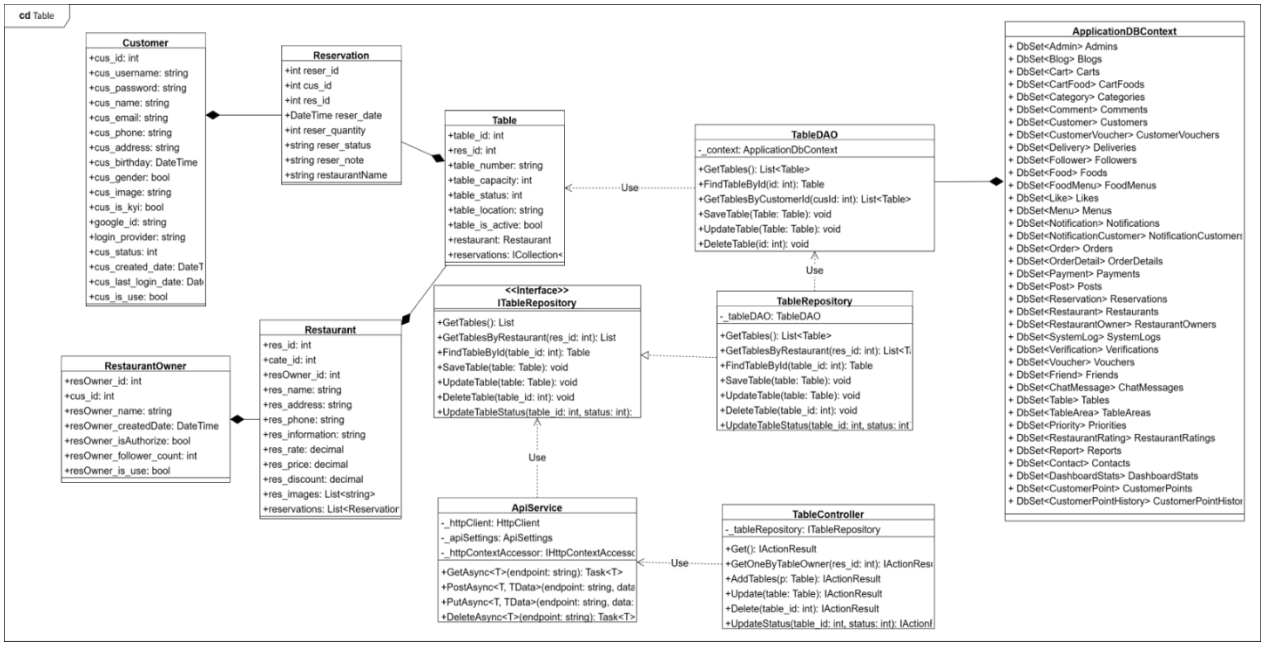


Figure 198: Class diagram of table

b) Sequence Diagram

View list of tables:

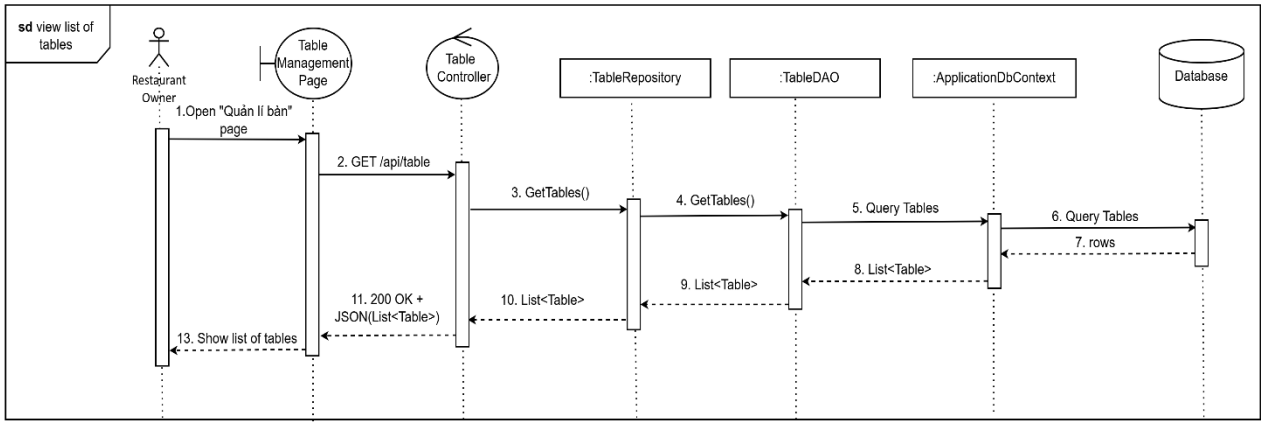


Figure 199: Sequence diagram view list of tables (Restaurant Owner)

Create table:

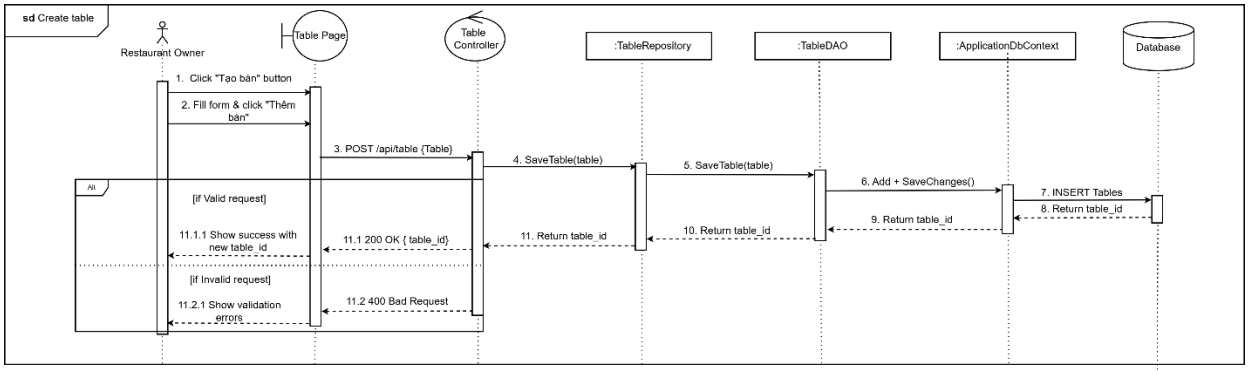


Figure 200: Sequence diagram create table (Restaurant Owner)

Modify table:

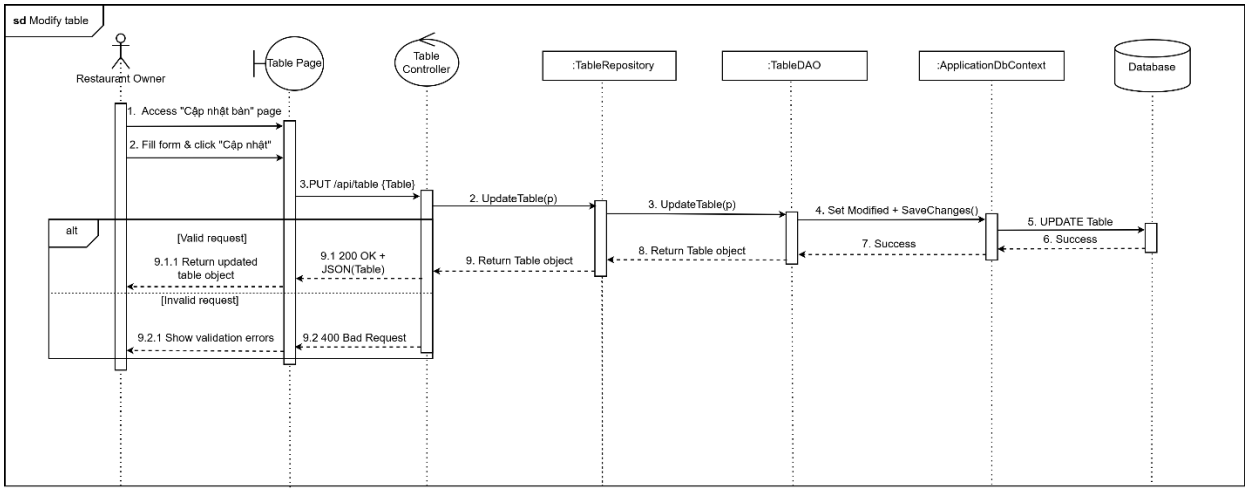


Figure 201: Sequence diagram modify table (Restaurant Owner)

Delete table:

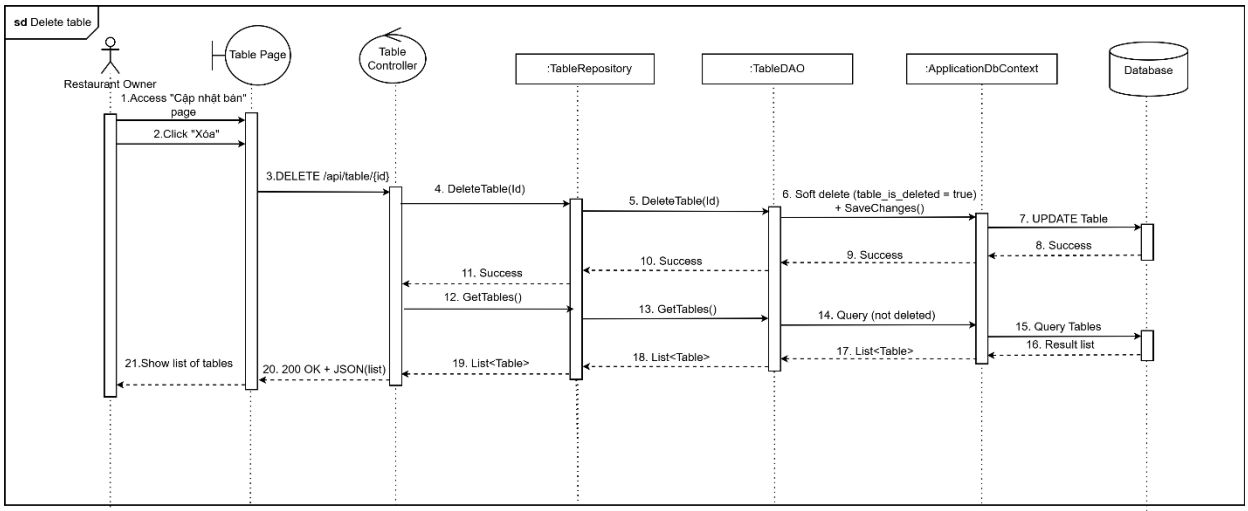


Figure 202: Sequence diagram delete table (Restaurant Owner)

3.10 AI Assistant Module

3.10.1 Runtime Inference Flow

Objective:

This module is responsible for analyzing user input to suggest relevant food-related tags using a trained DistilBERT model.

The base model used for fine-tuning is **distilbert-base-uncased**[17], an open-source pretrained language model provided by Hugging Face. It serves as the foundation for multi-label classification of food-related intents.

Additionally, the module includes a response generation component based on **VietAI/vit5-base**[18], a Vietnamese pretrained T5 model fine-tuned to produce descriptive food suggestions.

Functional Flow:

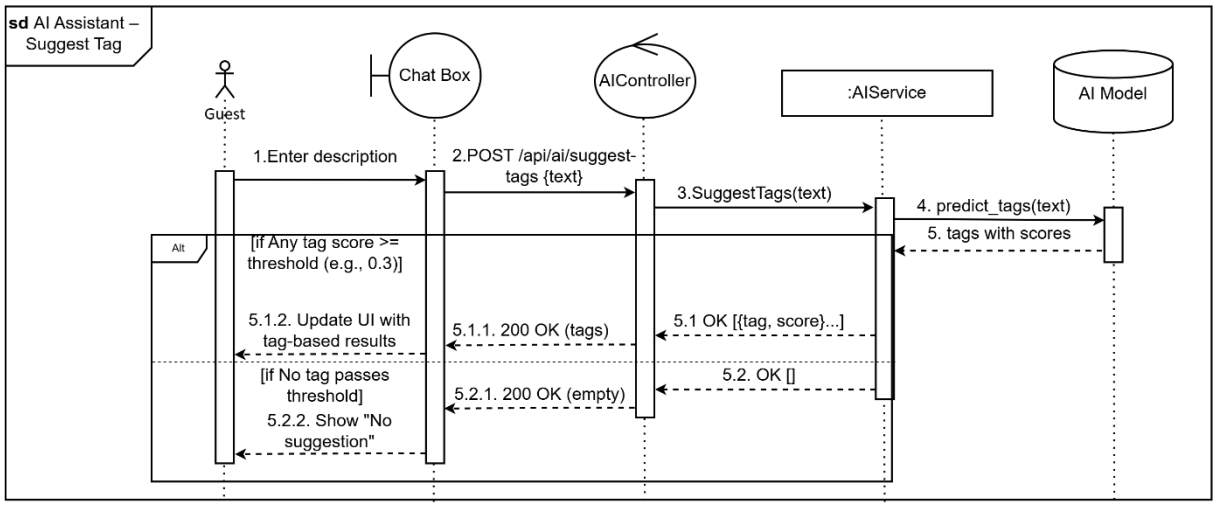


Figure 203: Sequence diagram AI assistant module - Suggest Tag (Guest)

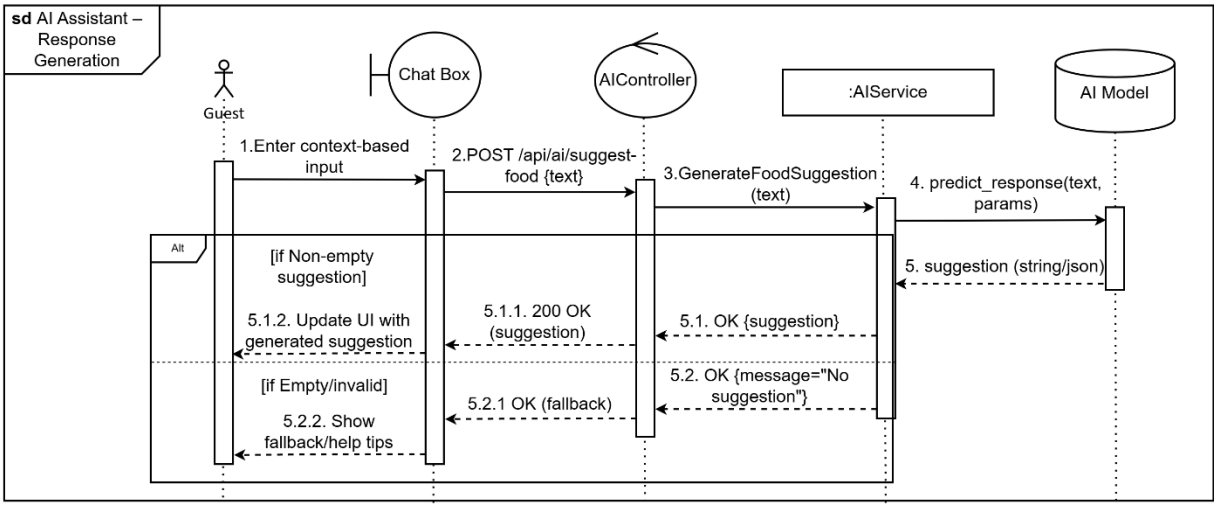


Figure 204: Sequence diagram AI assistant module - Response Generation (User)

[17] **Hugging Face – DistilBERT Base Uncased**: An open-source pretrained language model based on the Transformer architecture, optimized for speed and performance. Available at: distilbert/distilbert-base-uncased · Hugging Face

[18] **Hugging Face – VietAI VIT5 Base**: A Vietnamese pretrained sequence-to-sequence T5 model developed by VietAI, designed for text generation and understanding tasks. Available at: VietAI/vit5-base · Hugging Face

Suggest Tag (DistilBERT):

1. User enters a description (e.g., “I want spicy fried food”) in the web interface.
2. Client Controller (e.g.) calls IAIService.GetTagsFromText(string text).
3. AIService sends an HTTP request to API /api/AI/suggest-tags?text=....
4. API Controller (AIController.cs) receives the request and forwards it to IAIPredictRepository.
5. Repository loads the trained DistilBERT model and executes predict_tags(text).
6. The model returns predicted tags (those with probability ≥ 0.3, a threshold empirically chosen to balance precision and recall).
7. The API response is propagated back to the Client Service, then to the Client Controller, and finally shown on the UI.

Response Generation (VIT5-based):

In addition to tag prediction, the system uses a separate model to generate descriptive food suggestions in natural language based on user input.

1. The user enters a context-based input (e.g., “I feel tired, suggest something light”).
2. Client sends a request to the API endpoint /api/AI/suggest-food?text=....
   AIController forwards the request to the method GenerateFoodSuggestionAsync(...) in AIPredictRepository.
3. The repository constructs a task-specific prompt and calls the Python script predict_response.py using a trained VietAI/vit5-base model.
4. The model generates a contextual food description (e.g., "Món nước thanh mát, dễ tiêu, ít dầu mỡ").
5. The API returns this suggestion to the frontend for display to the user.

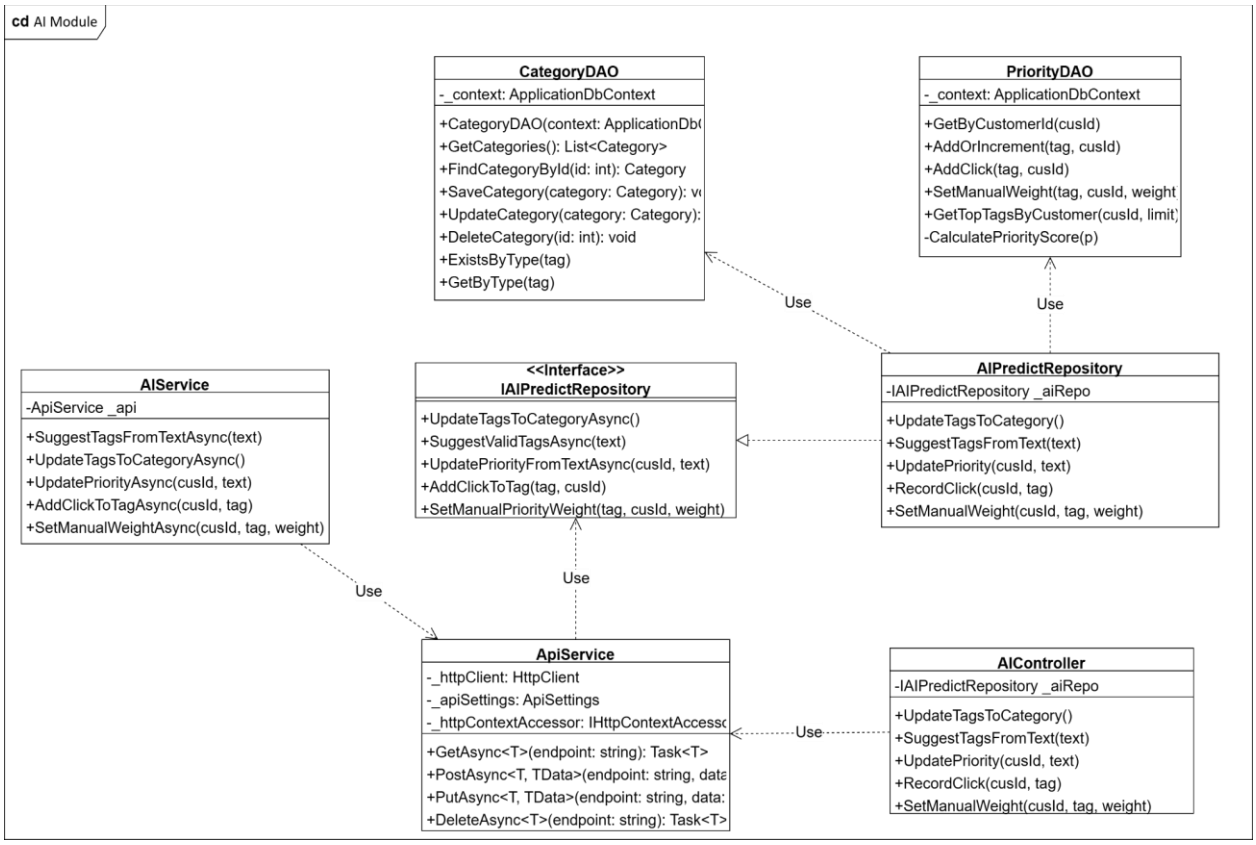


Figure 205: Class diagram AI assistant module

Explain and introduce the formula:
When the API endpoint /api/AI/priority/update is called, tags extracted from the AI model are used to update the customer's tag preferences.
Each tag's priority is updated using a scoring function implemented in the PriorityDAO, which combines multiple behavioral signals.
The formula used to calculate the score is:

$$score = (count \times 2) + (click_{count} \times 0.6) + decay + weight_{manual}$$

where:

- **count** is how often the tag was suggested.
- **click_count** reflects user interaction with the tag.
- **decay** = 1 / (1 + days_since_last_used) models recency of activity.
- **weight_manual** is a custom preference set by the user or admin.

This scoring strategy was designed by the development team based on domain knowledge and empirical testing to balance AI suggestions, user actions, and personalization.

3.10.2 Tag Prediction Model (DistilBERT)

This subsection outlines the training process used to develop the AI model responsible for food tag prediction based on natural language descriptions.
Natural Language Processing (NLP) is a field of artificial intelligence that focuses on enabling machines to understand, interpret, and generate human language. In this system, NLP techniques are applied to analyze user-provided descriptions of food preferences and extract meaningful tags that represent culinary features or styles.
This prediction task follows the supervised learning paradigm, where the model learns from input-output pairs (descriptions and tags) to minimize error during training.
Model Overview:

- **Model Type**: DistilBERTForSequenceClassification
- **Architecture**: Transformer-based encoder (DistilBERT)
- **Task**: Multi-label text classification
- **Problem Type**: multi_label_classification
- **Framework**: HuggingFace Transformers

Dataset Detail:
The dataset used for training is stored in the file food_descriptions_20k.csv, which was compiled from publicly available food review sources and curated internal examples. It contains pairs of user-provided natural language descriptions and their associated food-related tags, used for multi-label classification.

- **File:** food_descriptions_20k.csv [3]
- **Number of Samples:** ~80,000
- **Fields:**
  - **Text (string):** natural language description of user food preferences
  - **Label (string)**: comma-separated list of tags
- **Label Encoding:**
  Tags are mapped to indices using labels.json and encoded as binary multi-hot vectors for training.
- **Resource of Dataset:**
  The dataset was constructed from multiple publicly available sources and synthetic data generation. Specifically:
  - Publicly available examples from general food review datasets and open web resources were referenced to guide tag distribution and sentence patterns.

- AI-assisted generation using [19]ChatGPT to produce diverse and contextually relevant food descriptions aligned with the tag taxonomy of the system.
- Custom synthetic entries manually designed by the development team to ensure coverage across all predefined food tags, reflecting real-world use cases of the application.

No proprietary or copyrighted datasets were used. The dataset was entirely curated, expanded, and structured to match the requirements of the DineGO food-tagging system.

```
text,label
Tui mà đói là nghĩ ngay đến món rau trộn và ăn kèm cơm.,"mon_goi,mon_chinh"
Món vừa thổi vừa ăn này vừa ngon vừa dễ làm.,mon_nong
Đây là món ăn kèm cơm và bánh bèo ai cũng mê.,"mon_chinh,mon_mien_trung"
"Bạn thử món không thịt đi, ghiền luôn á.",mon_chay
"Bạn thử món gỏi cuốn và cao lầu và bữa chính đi, ghiền luôn á.","mon_goi,mon_mien_trung,mon_chinh"
Phải thử món thanh đạm và mới nấu một lần trong đời.,"mon_an_nhe,mon_nong"
Đây là món snack ai cũng mê.,mon_an_choi
Món cháo trắng này vừa ngon vừa dễ làm.,mon_sang
Tối nay làm đĩa ấm bụng là hết sẩy.,mon_nong
Một món mát lạnh và phở bò ăn cực đã.,"mon_lanh,mon_nuoc"
Món này vị ngọt béo.,mon_ngot
Một món vừa thổi vừa ăn ăn cực đã.,mon_nong
Tui mà đói là nghĩ ngay đến món món phụ.,mon_diem_tam
Món này quán vỉa hè.,mon_binh_dan
Phải thử món món phụ một lần trong đời.,mon_diem_tam
Một món xôi mặn và trứng cút ăn cực đã.,"mon_sang,mon_trung"
```

Figure 206: data sample

Training Configuration:

Based on the dataset characteristics and the model's intended application, the training configuration was designed to balance computational efficiency with model performance. The parameters below were selected after empirical testing to ensure stable convergence without overfitting.

Table 19: Training strategy table

Parameter	**Value**	**Note**
Number of Epochs	3	The number of full passes through the training dataset
Batch Size	8	Number of samples processed per batch on each device
Gradient Accumulation	2 (effective batch size = 16)	Accumulates gradients to simulate a batch size of 16 (8 × 2)
Learning Rate	2e-5	Step size used by the optimizer to update model weight

[19] ChatGPT OpenAI: chatgpt.com

Parameter	Value	Note
Loss Function	BCEWithLogitsLoss	Suitable for multi-label classification tasks
Tokenizer	DistilBertTokenizerFast	Fast WordPiece tokenizer used to process input text
Max Sequence Length	128 tokens	Maximum number of tokens per input sequence
Optimizer	AdamW (default)	Adaptive optimizer with weight decay, widely used for Transformer models
Mixed Precision (fp16)	Enabled if CUDA available	Reduces memory usage and speeds up training on supported GPUs
Save Strategy	Every epoch	Model checkpoints are saved after each epoch
Logging Strategy	Every epoch	Training loss and metrics are logged once per epoch
Output Directory	../models/intent-food-model	Folder path where model checkpoints are saved

Training Pipeline:

The training procedure is implemented in the script train_unified.py and follows the following steps:

1. **Data Preprocessing**
    - Load the dataset using Pandas
    - Encode labels into multi-hot vectors
    - Convert into HuggingFace Dataset format
2. **Tokenization**
    - Tokenize the input texts using DistilBertTokenizerFast
3. **Dataset Mapping**
    - Add tokenized input and label tensors to the dataset
4. **Model Initialization**
    - Load pre-trained DistilBERT model
    - Set number of output labels and problem type
5. **Training Execution**
    - Use Trainer API with defined TrainingArguments
    - Monitor training loss and save checkpoints
6. **Model Saving**
    - Save trained model, tokenizer, and label mapping to disk

Inference Flow:

Once trained, the model is loaded by the backend system and used for inference via the predict_tags(text) function. It returns all tags whose predicted probability exceeds a threshold (typically 0.3).

This configuration ensures efficient adaptation of a general-purpose language model to the domain-specific task of food tag classification, maintaining performance while avoiding overfitting.

Result Example:

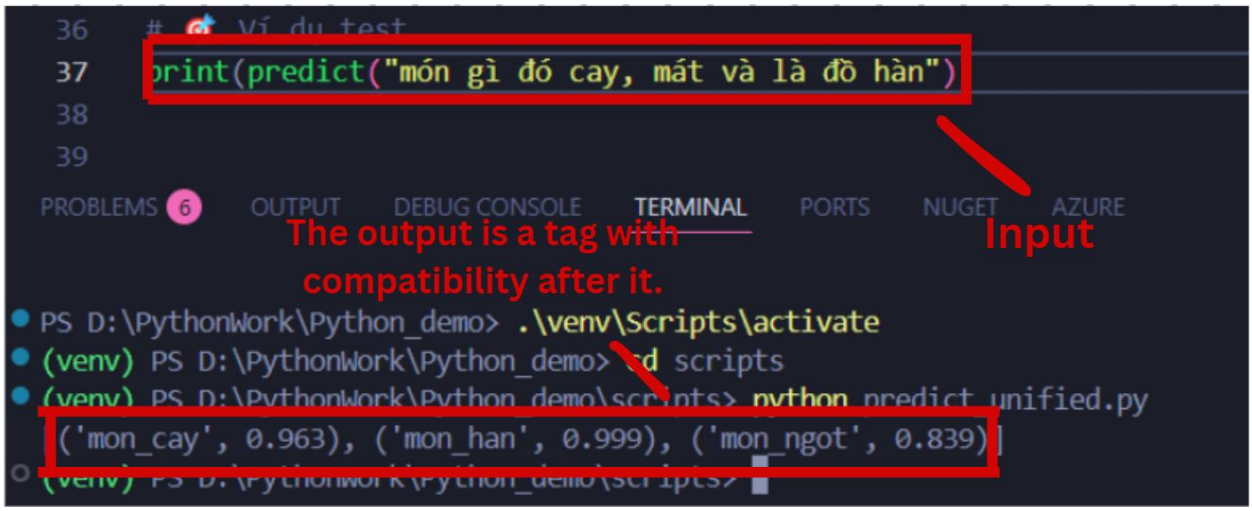


Figure 207: result

Training Process Video:

A screen-recorded video demonstrating the full model training process (including data loading, preprocessing, training configuration, and model saving) is available at the following link:

[20]Training Process Recording.

3.10.3 Response Generation(VIT5)

This subsection outlines the training process used to develop the **response generation model**, which produces natural-language suggestions about food based on user context.

Similar to the tag prediction task, this module also leverages **Natural Language Processing (NLP)** techniques to interpret user input. However, instead of classifying it into tags, the model is trained to generate coherent and relevant textual responses.

The model is fine-tuned under a **supervised learning paradigm**, using input-output pairs (prompt, expected response) to learn how to produce meaningful suggestions in Vietnamese. The underlying architecture is sequence-to-sequence (seq2seq), suitable for tasks that require transforming one piece of text into another.

Objective:

This module is responsible for generating descriptive food suggestions based on user context using a fine-tuned Vietnamese T5 model (VietAI/vit5-base). Unlike tag prediction, which classifies user input into structured categories, this module focuses on producing free-form, natural language outputs.

Model Overview:

- **Model Type:** Sequence-to-Sequence (T5 architecture)

[20] Training Process Recording. Available at: video

- **Base Model:** VietAI/vit5-base (Hugging Face)
- **Task:** Text generation (response generation)
- **Framework:** HuggingFace Transformers
- **Input Format:** Prompt (e.g., "Suggest a dish for...")
- **Output:** Descriptive suggestion (text response)

Dataset Detail:

The dataset used for fine-tuning the response generation model is stored in the file prompt_response_15k_concrete.csv. This dataset was internally curated to simulate natural food-related conversation patterns. Each sample includes a prompt that reflects a user's food-related intent or emotional context, and a corresponding target that represents the desired model output — a descriptive suggestion for food.

- **File:** prompt_response_15k_concrete.csv [4]
- **Number of Samples:** ~15,000
- **Fields:**
  - **Prompt (string)**: input text simulating a user's intent or situation (e.g., *"Suggest a dish when someone is tired and it's hot outside."*)
  - **Target (string):** expected food-related response in natural Vietnamese (e.g., *"Món nước thanh mát, dễ tiêu hóa, ít dầu mỡ."*)
- **Resource of Dataset:**

  The dataset was constructed from multiple publicly available sources and synthetic data generation. Specifically:
  - Publicly available examples from general food review datasets and open web resources were referenced to guide tag distribution and sentence patterns.
  - AI-assisted generation using [21]ChatGPT to produce diverse and contextually relevant food descriptions aligned with the tag taxonomy of the system.
  - Custom synthetic entries manually designed by the development team to ensure coverage across all predefined food tags, reflecting real-world use cases of the application.

  No proprietary or copyrighted datasets were used. The dataset was entirely curated, expanded, and structured to match the requirements of the DineGO food-tagging system.

DataSet Sample

```
prompt,target
"Phản hồi nhẹ nhàng cho người dùng nói: thiệt là, tui mới ăn xong",Hi vọng bạn sẽ thấy vui hơn nha.<br>
Phản hồi nhẹ nhàng cho người dùng nói: tui chỉ muốn được nghe gợi ý nhẹ nhàng,Bạn xứng đáng được nghỉ ngơi mà.<br>
Phản hồi nhẹ nhàng cho người dùng nói: tui hơi buồn hôm nay,Hi vọng bạn sẽ thấy vui hơn nha.<br>
"Phản hồi nhẹ nhàng cho người dùng nói: thiệt là, cảm thấy có chút buồn chán",Nghe bạn nói là tui cũng vui theo luôn á.<br>
Phản hồi nhẹ nhàng cho người dùng nói: hihi tui thèm gì đó mà không biết là gì,Bạn xứng đáng được nghỉ ngơi mà.<br>
"Phản hồi nhẹ nhàng cho người dùng nói: hihi, tớ đang phân vân ăn gì",Vậy là bụng bạn đã no rồi đó ha.<br>
"Phản hồi nhẹ nhàng cho người dùng nói: thiệt là, cảm thấy có chút buồn chán",Tui ở đây để lắng nghe bạn nè.<br>
Sao mọi người ai cũng bận quá vậy ta,"Mọi chuyện sẽ ổn thôi, tin tui đi.<br>"
Haha đúng gu tui luôn,"Vui vẻ lên nha, có tui ở đây rồi.<br>"
Tui đang nghĩ về tương lai,Tui ở đây để lắng nghe bạn á.<br>
Tui đang rảnh nè,Haha bạn dễ thương ghê.<br>
Bạn đang làm gì vậy?,"Chào bạn, có gì mới không?<br>"
Chào bạn nha,Thỉnh thoảng như vậy cũng tốt mà.<br>
Tui đang rảnh nè,"Bạn cứ tâm sự nha, tui không chê đâu.<br>"
Nay trời đẹp ha,Nghỉ ngơi chút cũng được mà.<br>
```

Figure 208: Sample training data for AI response generation

[21] ChatGPT OpenAI: chatgpt.com

This dataset enables supervised fine-tuning of a Vietnamese T5-based model (vit5-base) for the task of contextual response generation in the food recommendation domain.

Training Configuration:

Based on the dataset characteristics and the model's intended application, the training configuration was designed to balance computational efficiency with model performance. The parameters below were selected after empirical testing to ensure stable convergence without overfitting.

Parameter	Value	Note
Pretrained Model	VietAI/vit5-base	Vietnamese T5 variant pretrained on large corpora
Epochs	5	Full passes through dataset
Batch Size	4	Per-device batch size
Gradient Accumulation	2	Simulates batch size of 8
Max Sequence Length	64	Token length for both input and output
Learning Rate	5e-5	AdamW optimizer with default settings
Mixed Precision (fp16)	Enabled if CUDA available	Optimized training on GPU
Save Strategy	Every epoch	Saves model checkpoint after each epoch
Output Directory	../models/vit5-response-model	Final model storage location

Training Pipeline:

- **Step 1:** Load dataset from CSV using Pandas.
- **Step 2:** Tokenize prompt and target using AutoTokenizer.
- **Step 3:** Encode input-output pairs and map to HuggingFace Dataset.
- **Step 4:** Configure Seq2SeqTrainer with AutoModelForSeq2SeqLM.
- **Step 5:** Train the model and save it to disk for later use.

Inference Flow:

- The system constructs a prompt using build_prompt(input_text) before calling the model.

- The model is executed via predict_response.py, loaded from the directory ../models/vit5-response-model.
- A sample input like *"Trời nóng quá, nên ăn gì?"* will produce output like *"Món nước thanh nhẹ, dễ tiêu hóa"*.
- The API endpoint used is /api/AI/suggest-food.

Result Example

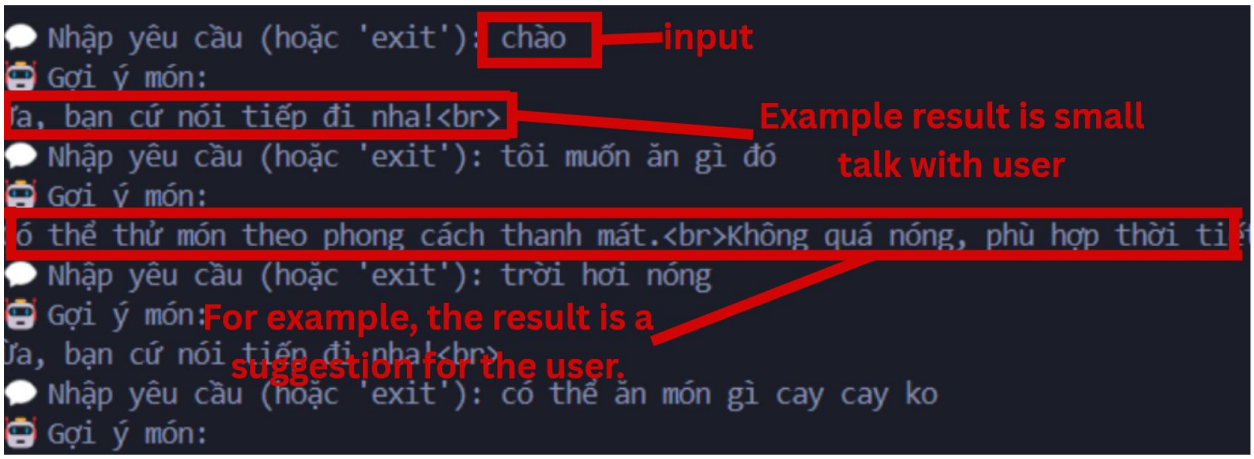


Figure 209: Example interaction and AI-generated food suggestion

Training Process Video:
A screen-recorded video demonstrating the full training process is available at this [22]video.

[22] Training Process Recording. Available at: video

3.11 Order

a) Class Diagram

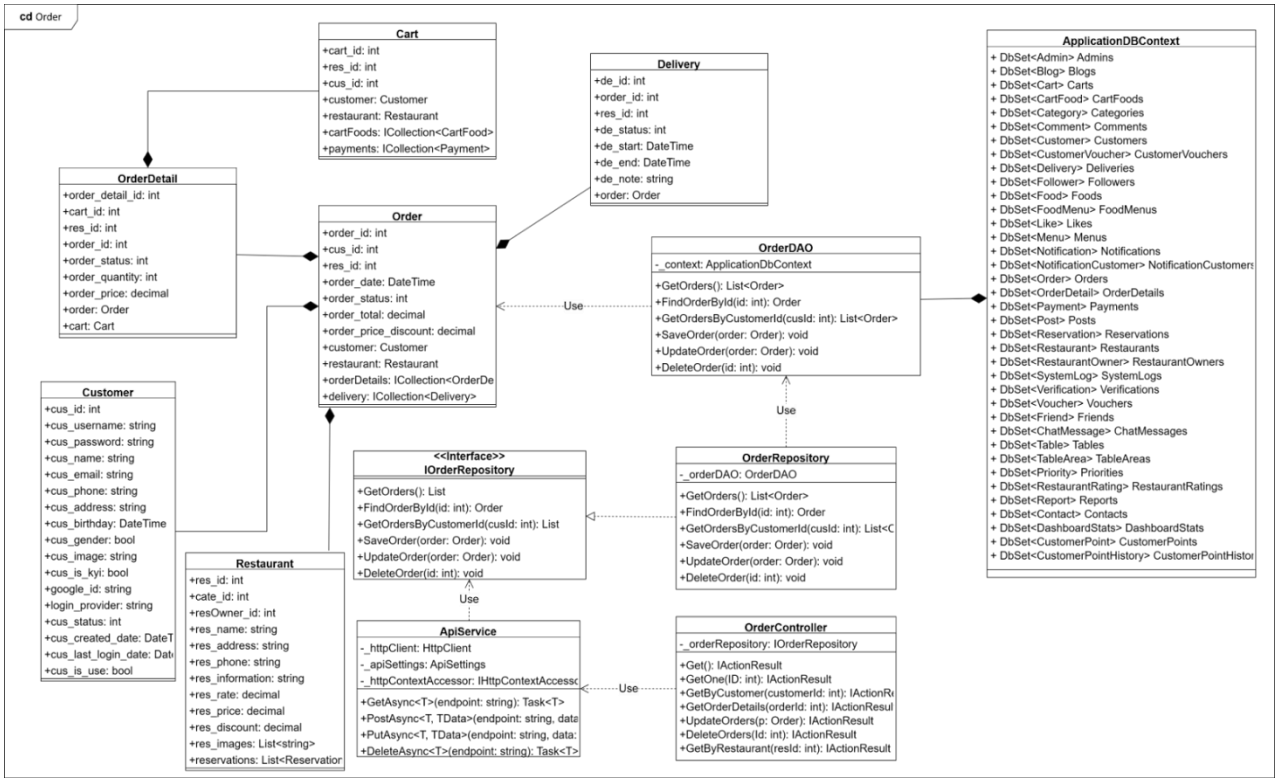


Figure 210: Class diagram for Order module

b) Sequence Diagram

View list of orders:

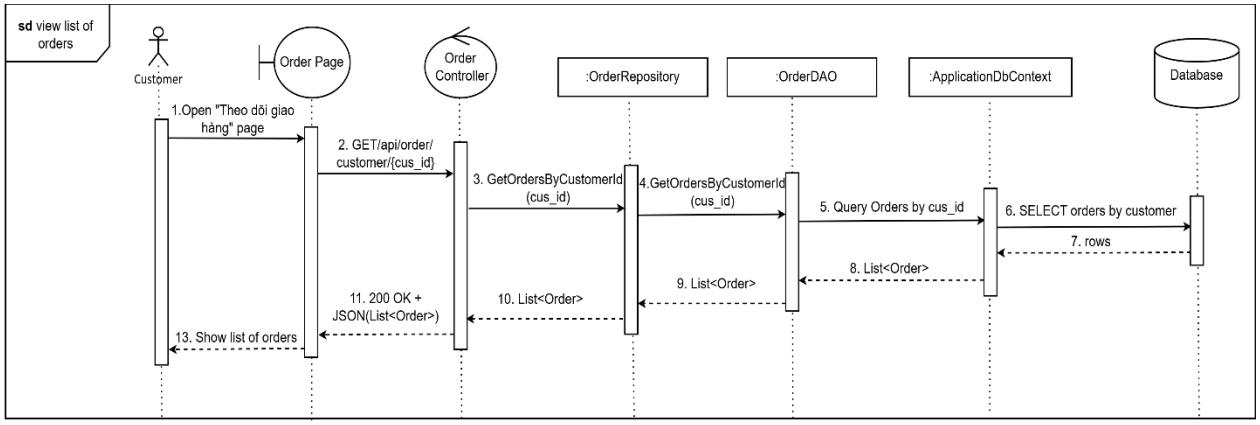


Figure 211: Sequence diagram view list of orders food (Customer)

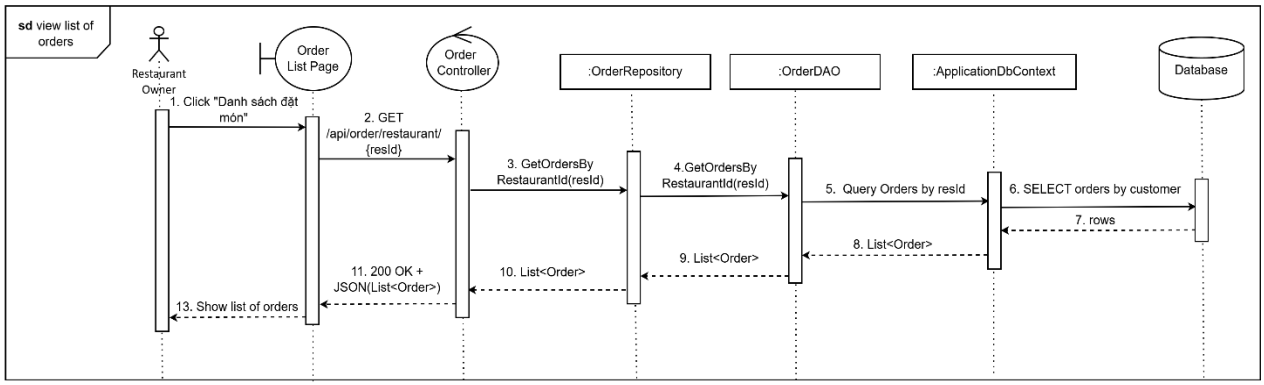


Figure 212: Sequence diagram view list of orders food (Restaurant Owner)

View order detail:

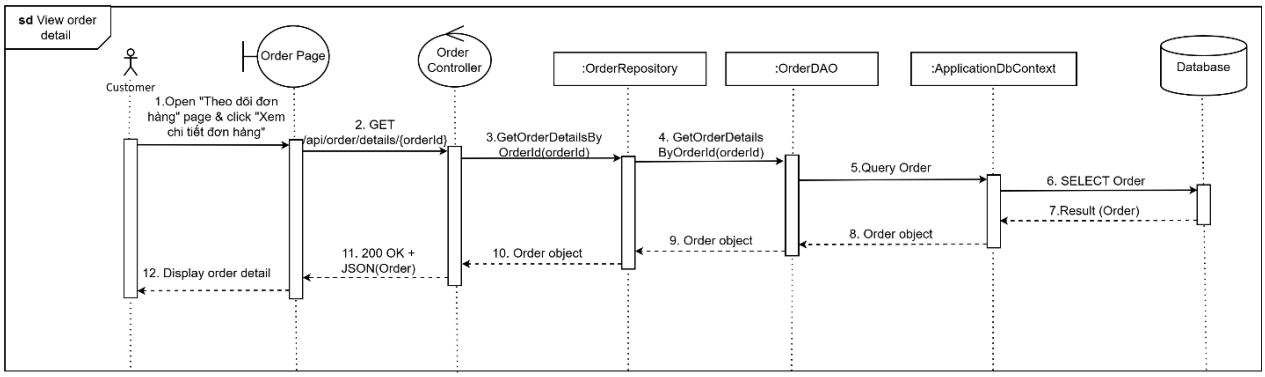


Figure 213: Sequence diagram view history order detail (Customer)

Create order:

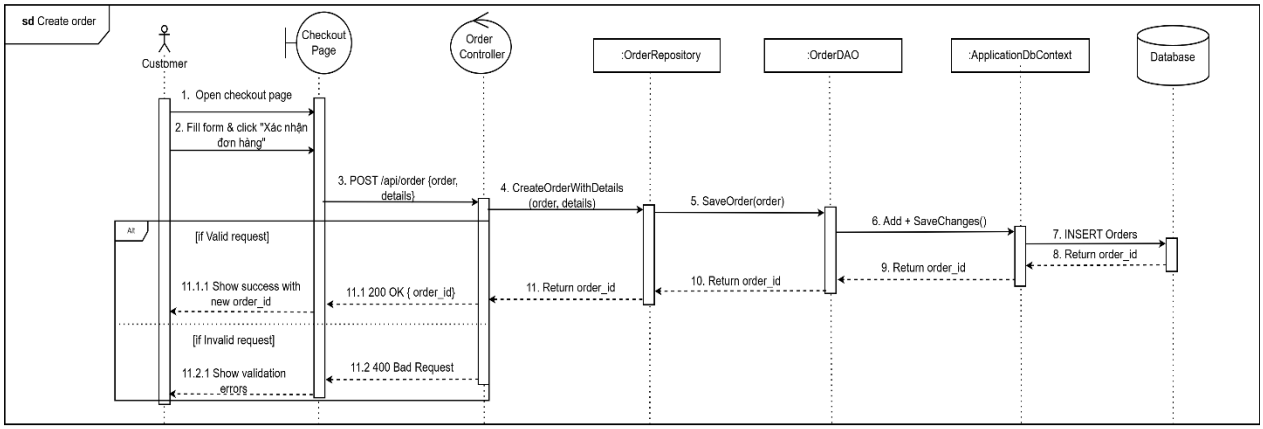


Figure 214: Sequence diagram create order(Customer)

Modify order:

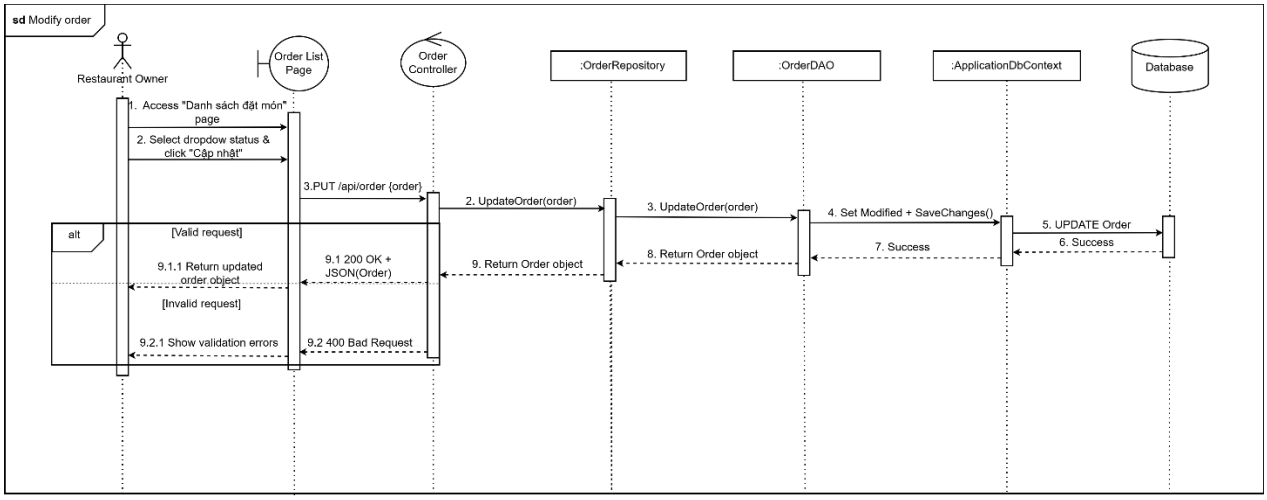


Figure 215: Sequence diagram modify order(Restaurant Owner)

3.12 Voucher

a) Class Diagram

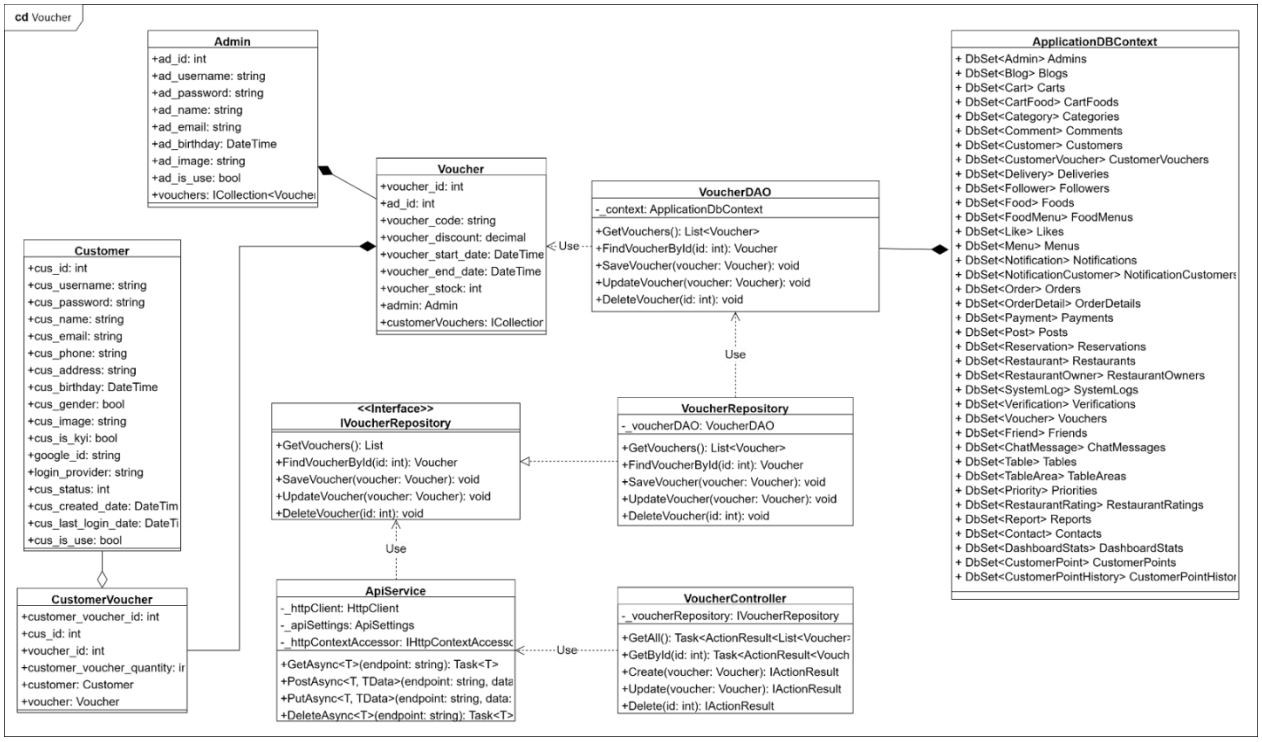


Figure 216: Class diagram voucher

b) Sequence Diagram

View list of vouchers:

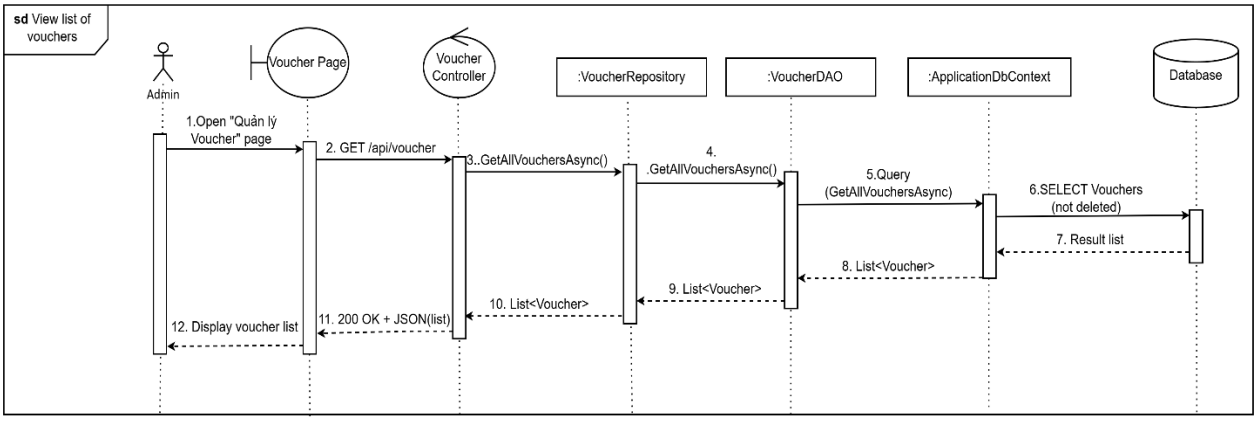


Figure 217: Sequence diagram view list of vouchers (Admin)

Create voucher:

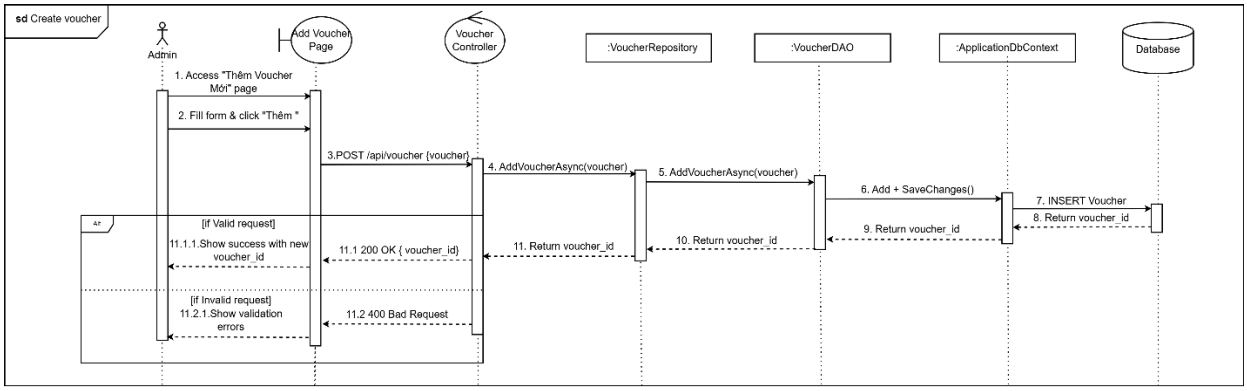


Figure 218: Sequence diagram create voucher (Admin)

Modify voucher:

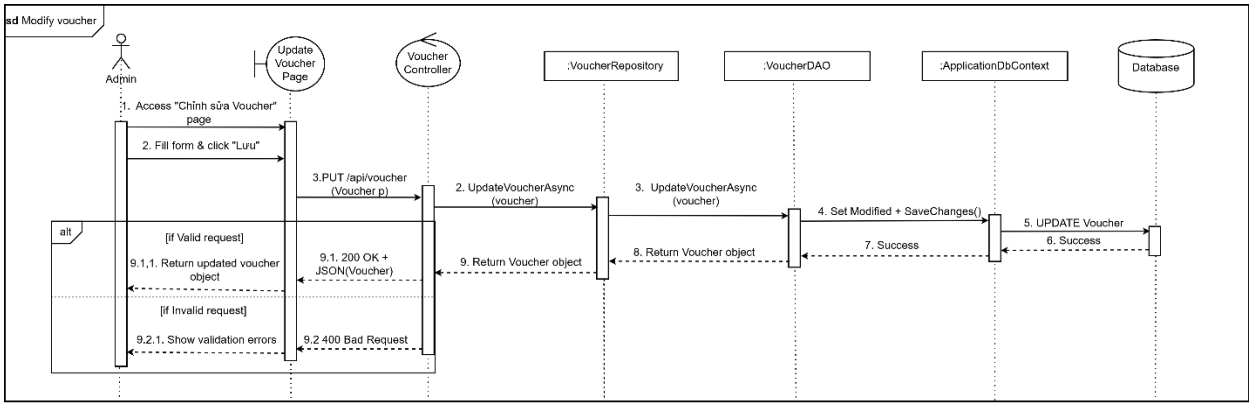


Figure 219: Sequence diagram modify voucher (Admin)

Delete voucher:

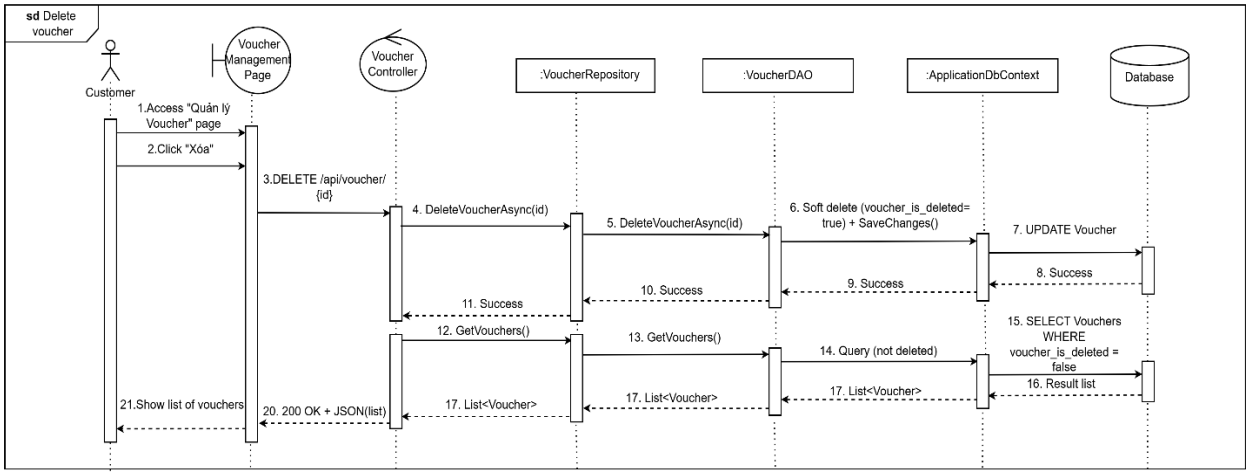


Figure 220: Sequence diagram delete voucher (Admin)

3.13 Verification

a) Class Diagram

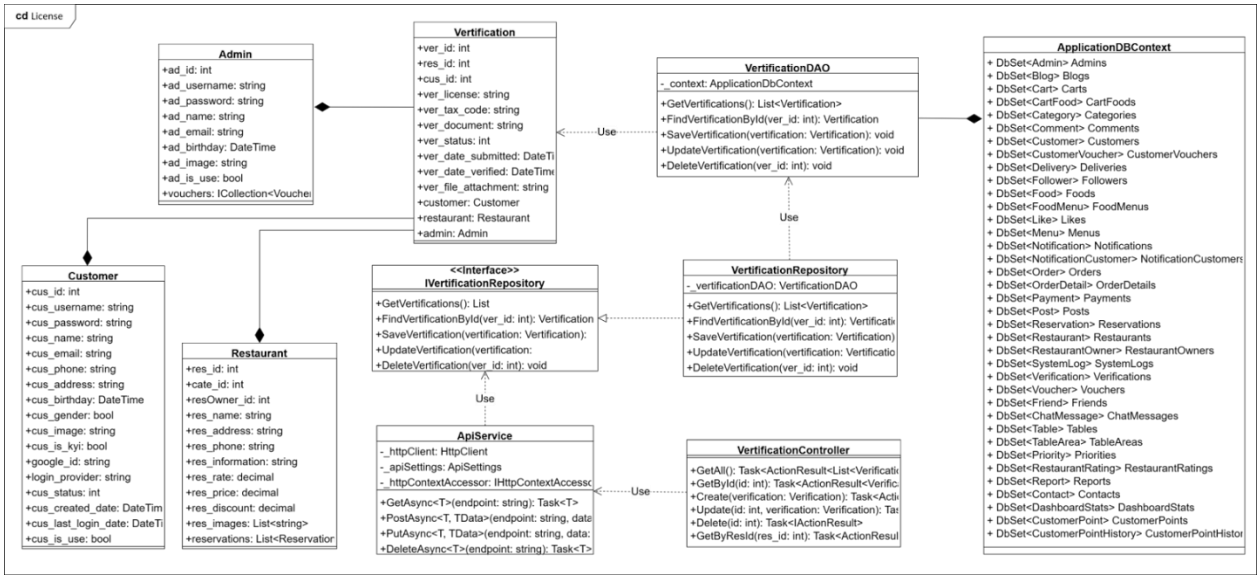


Figure 221: Class diagram verification

b) Sequence Diagram

View list of Verifications:

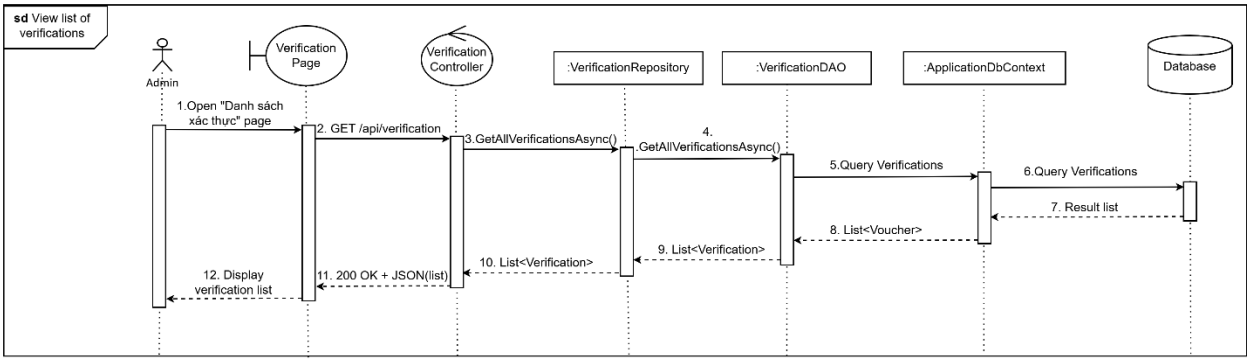


Figure 222: Sequence diagram view list of verifications (Admin)

Approve Verification:

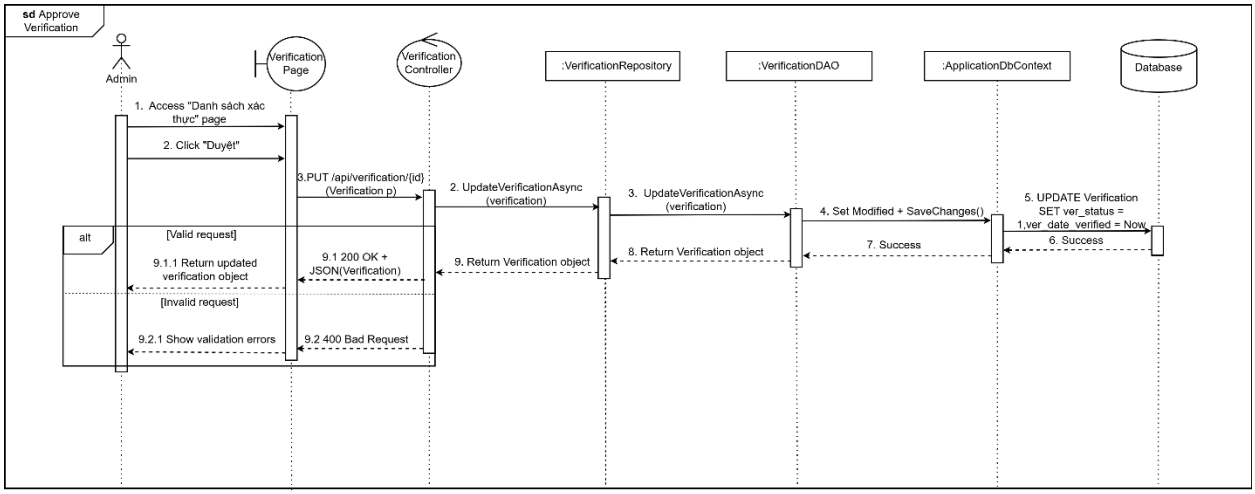


Figure 223: Sequence diagram approve verification (Admin)

Reject Verification:

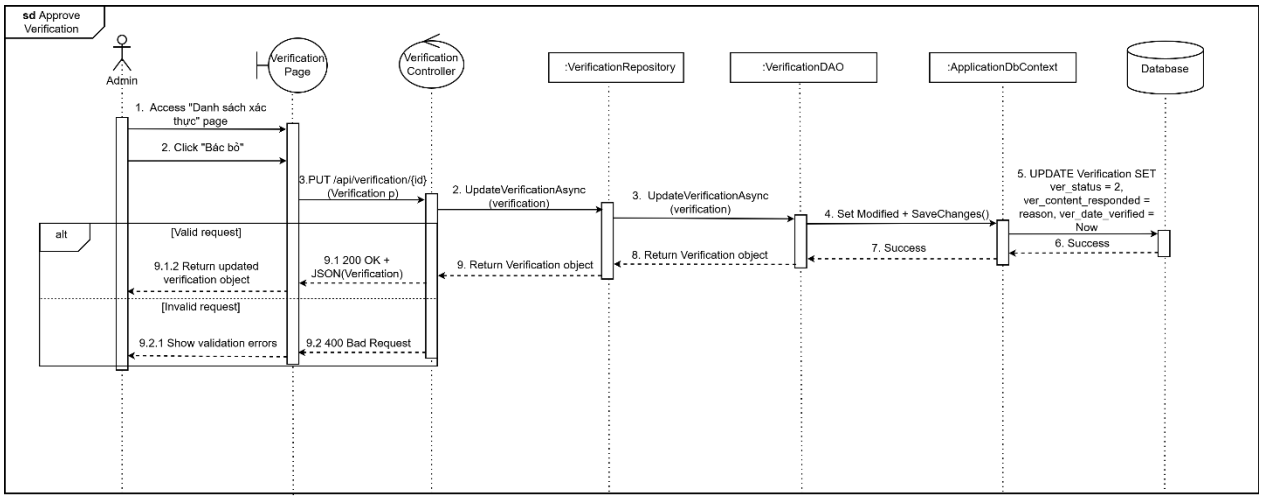


Figure 224: Sequence diagram reject verification (Admin)

3.14 Notification

a) Class Diagram

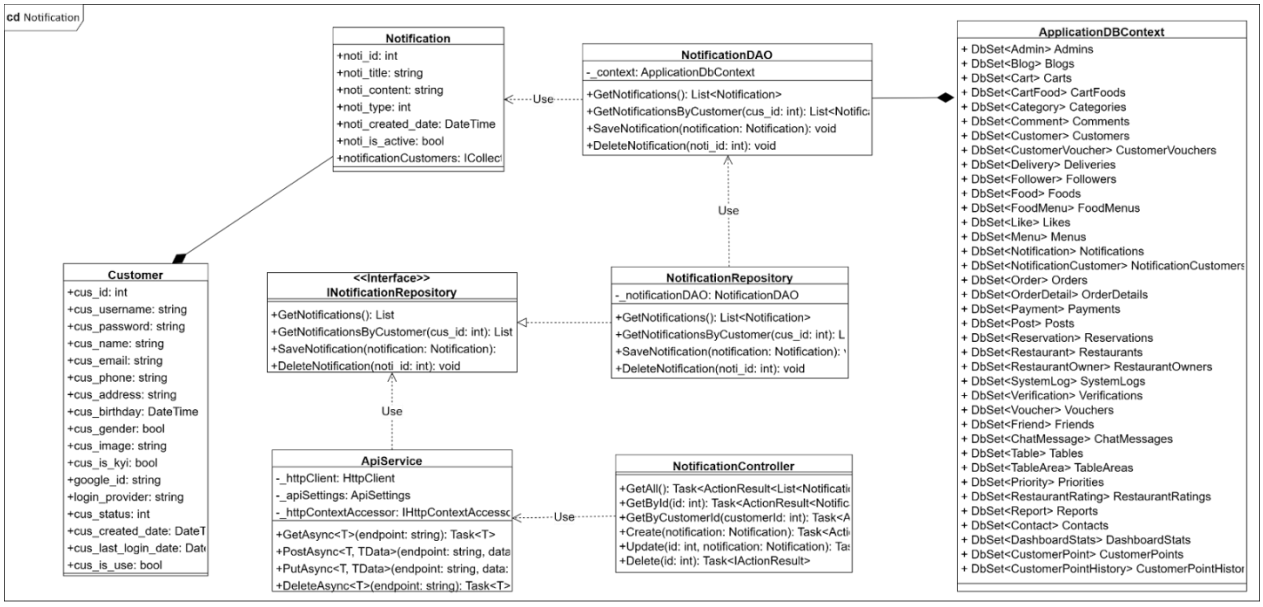


Figure 225: Class diagram notification

b) Sequence Diagram

View list notifications:

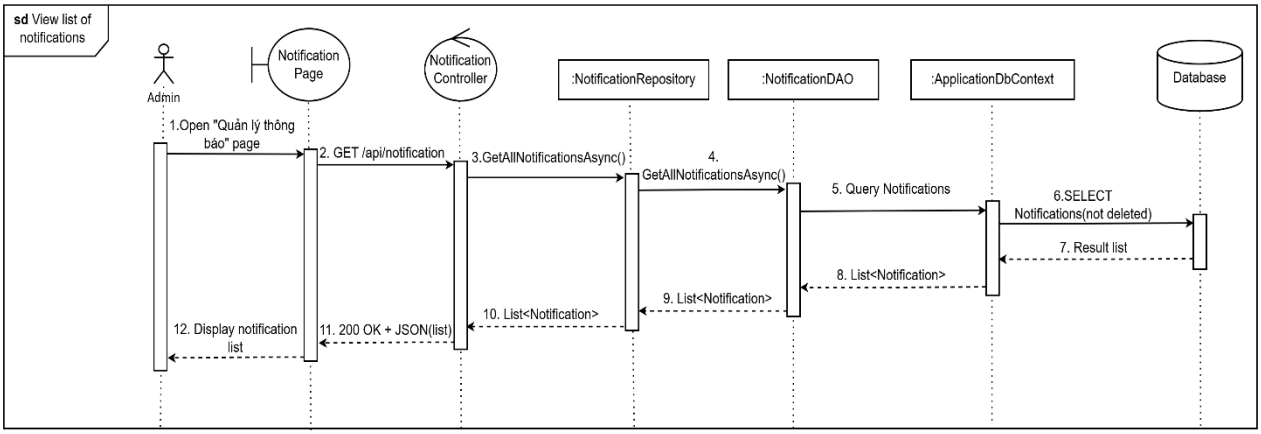


Figure 226: Sequence diagram view list notifications (Admin)

Create notification:

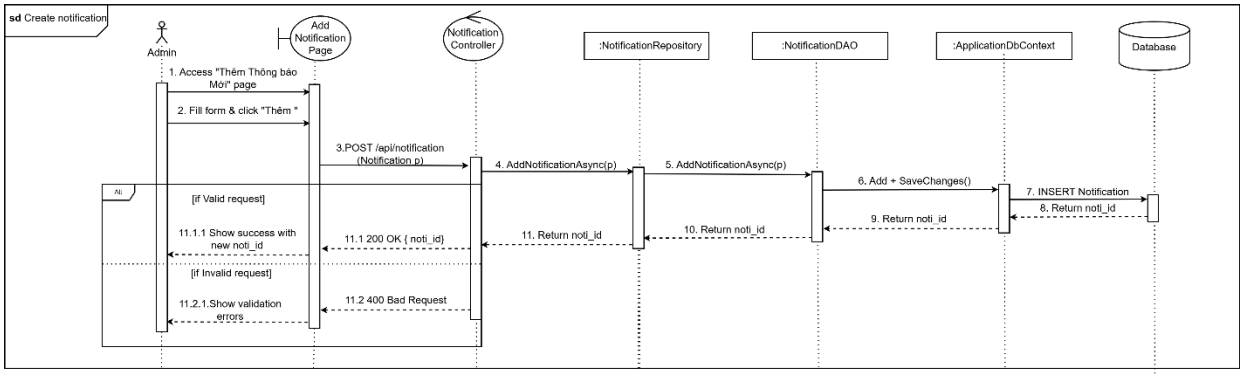


Figure 227: Sequence diagram create notification (Admin)

Modify notification:

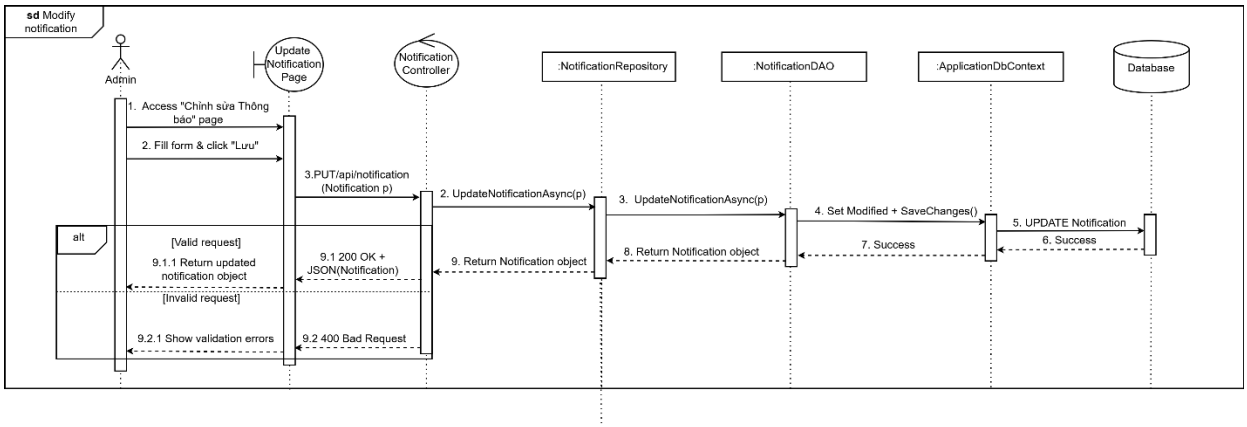


Figure 228: Sequence diagram modify notification (Admin)

Delete notification:

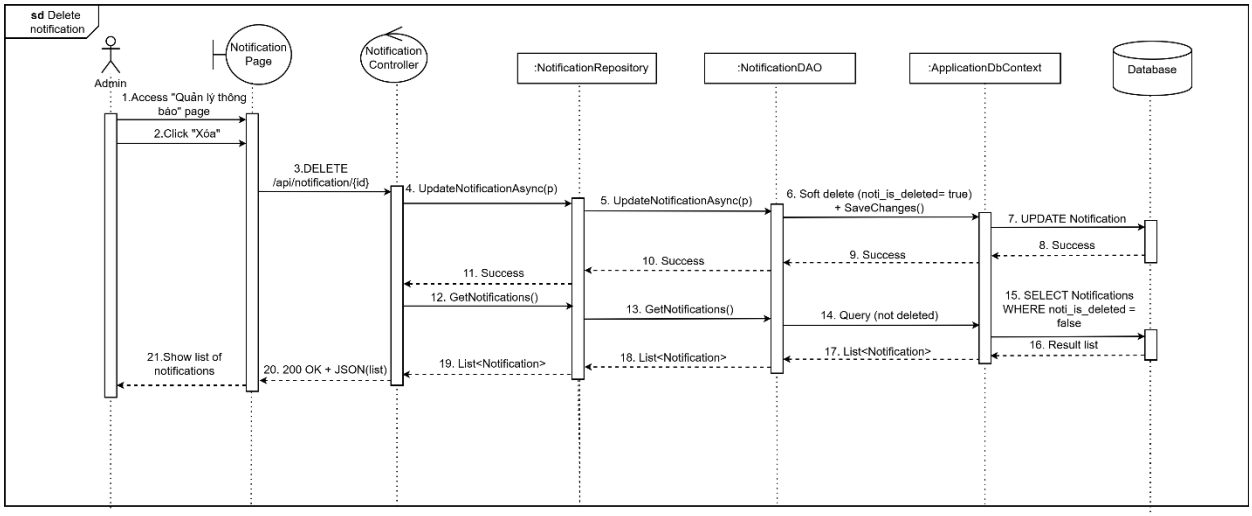


Figure 229: Sequence diagram delete notification (Admin)

3.15 Cart

a) Class Diagram

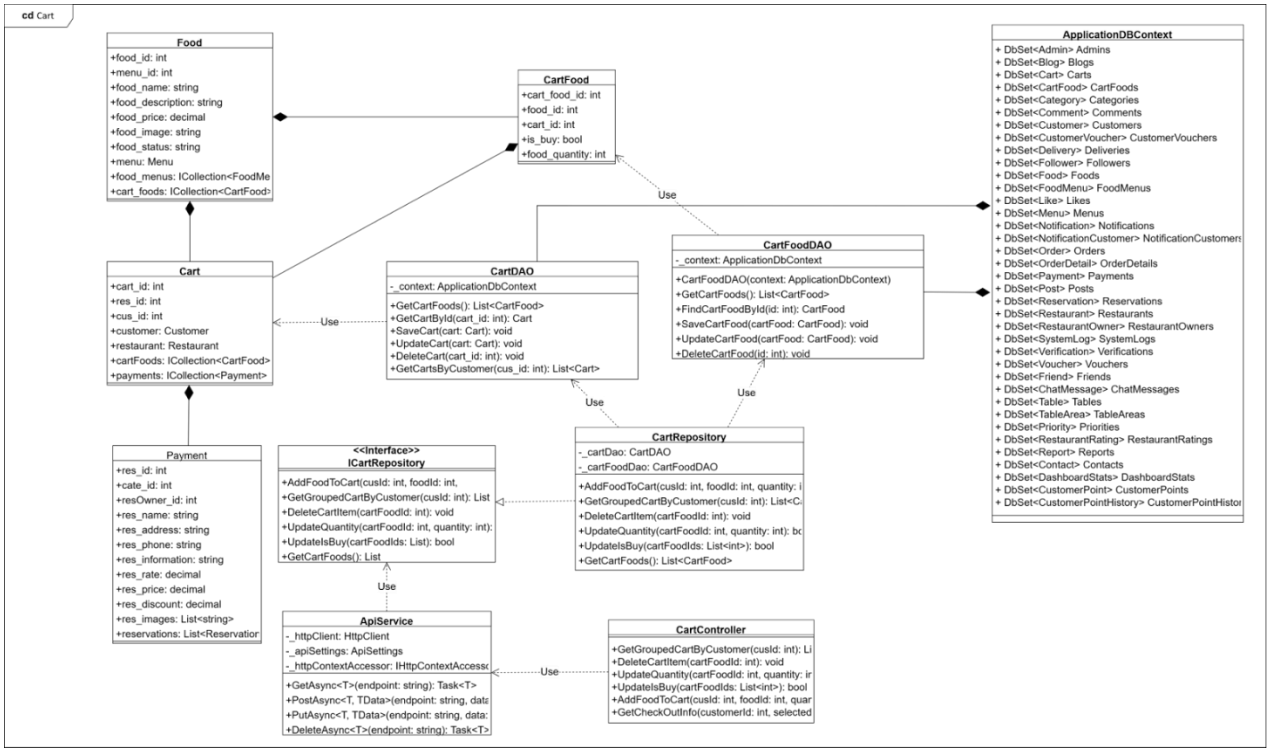


Figure 230: Class diagram cart

b) Sequence Diagram

View cart:

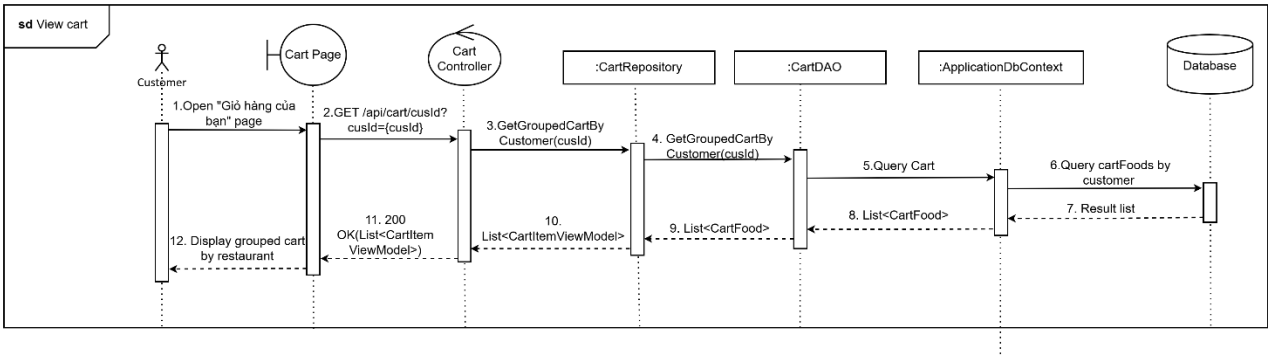


Figure 231: Sequence diagram view cart (Customer)

Add food to cart:

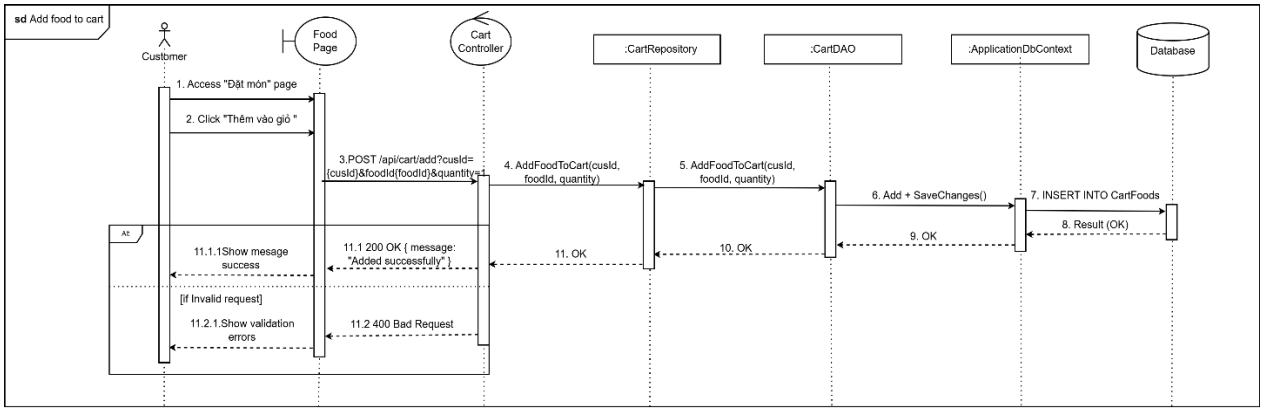


Figure 232: Sequence diagram add food to cart (Customer)

Modify cart item quantity:

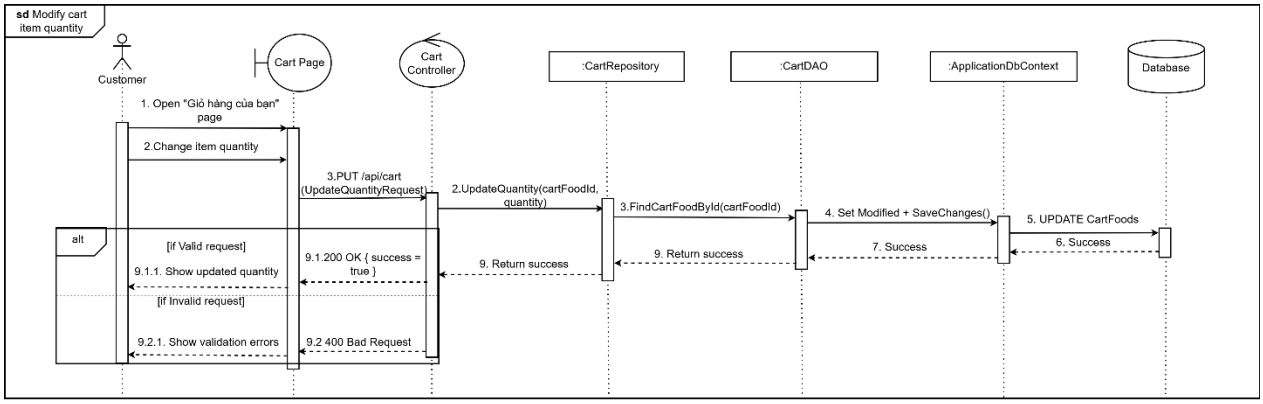


Figure 233: Sequence diagram modify cart item quantity (Customer)

Delete food in cart:

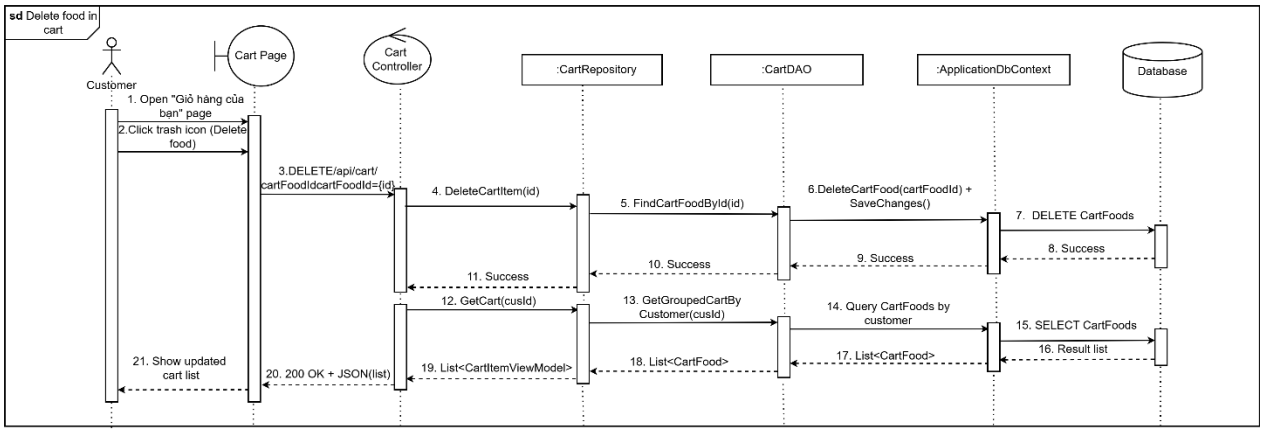


Figure 234: Sequence diagram Delete food in cart (Customer)

3.16 Menu

a) Class Diagram

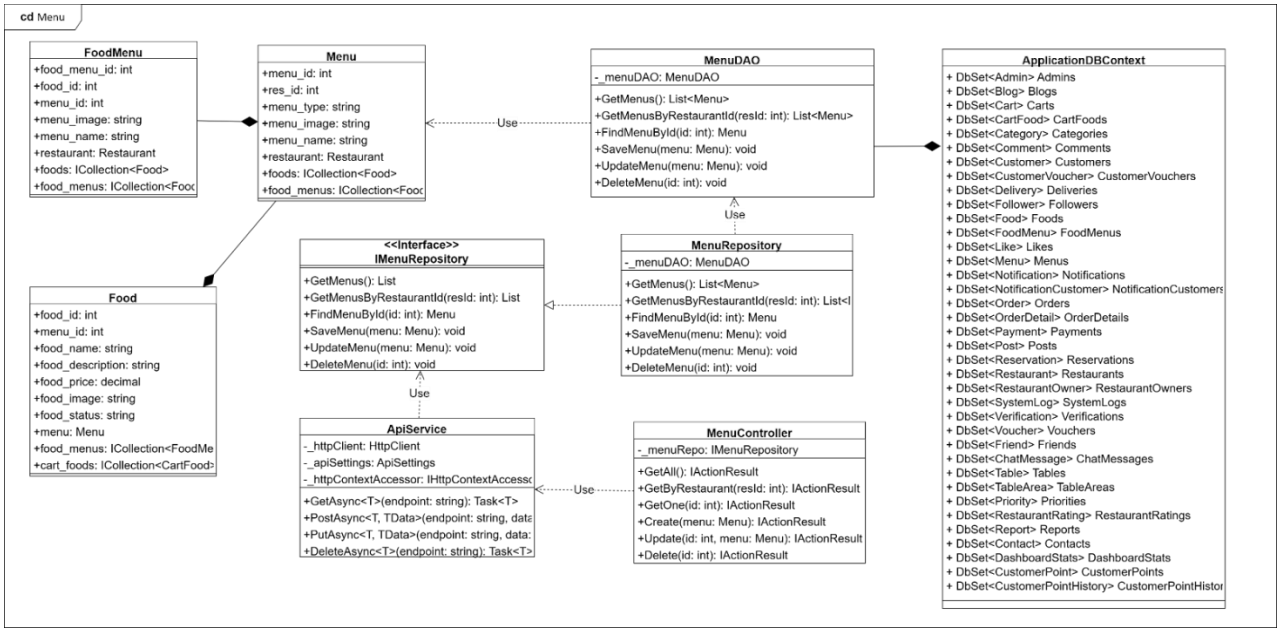


Figure 235: Class diagram of menu

b) Sequence Diagram

View list of menus by restaurant:

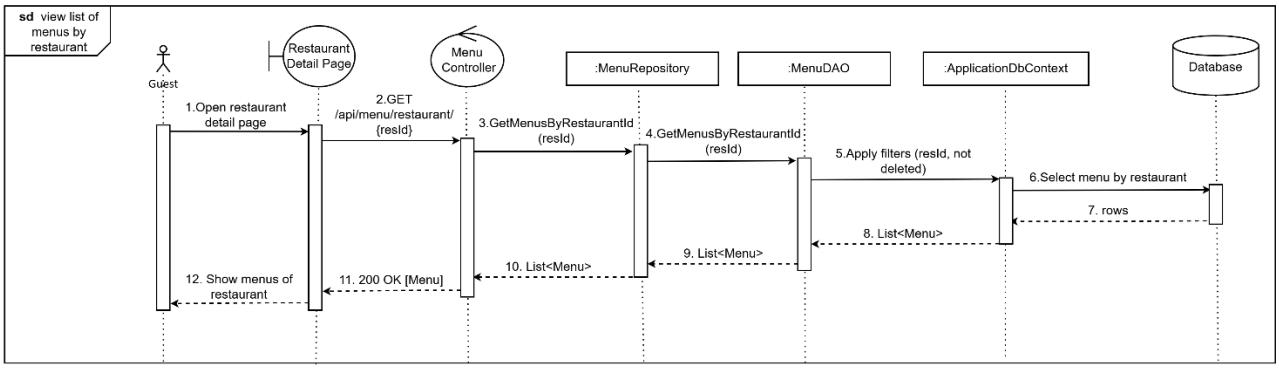


Figure 236: Sequence diagram view list of menus by restaurant (Guest)

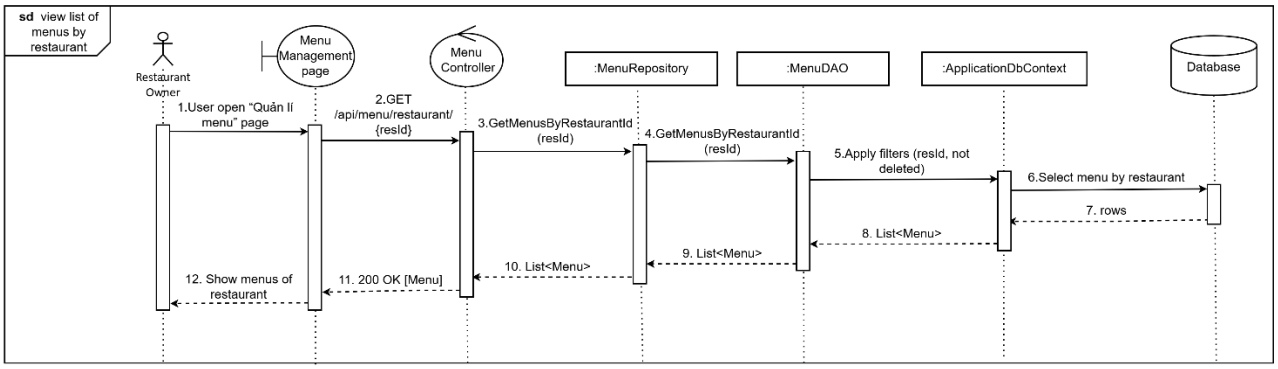


Figure 237: Sequence diagram view list of menus by restaurant (Restaurant Owner)

Create menu:

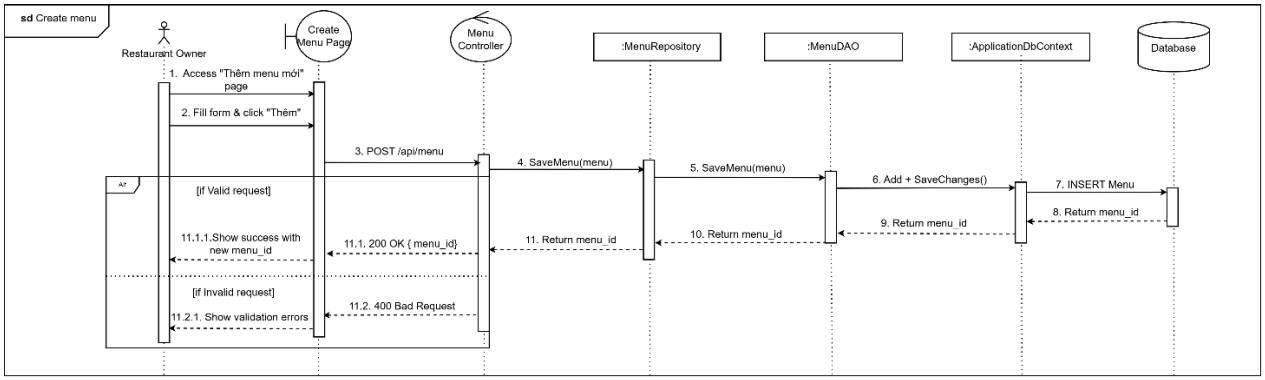


Figure 238: Sequence diagram create menu (Restaurant Owner)

Modify menu:

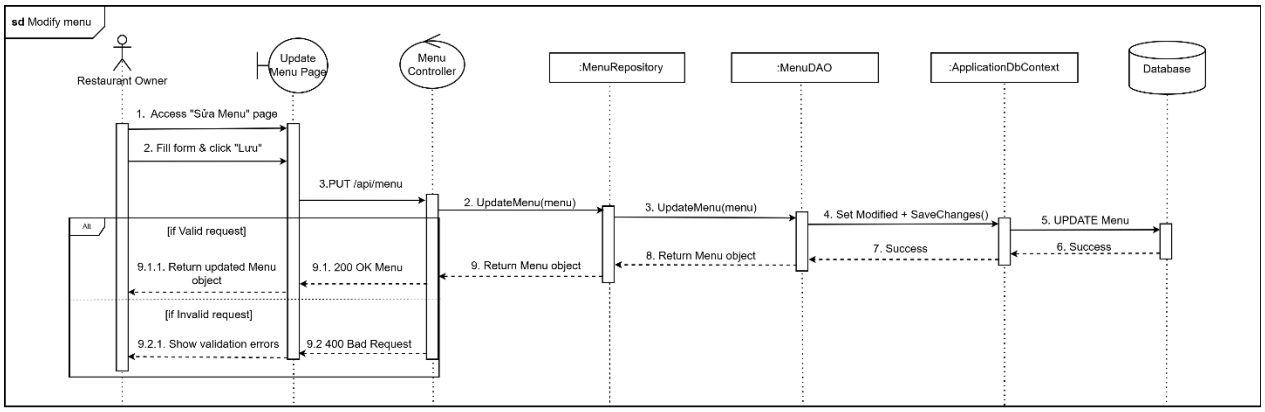


Figure 239: Sequence diagram modify menu (Restaurant Owner)

Delete menu:

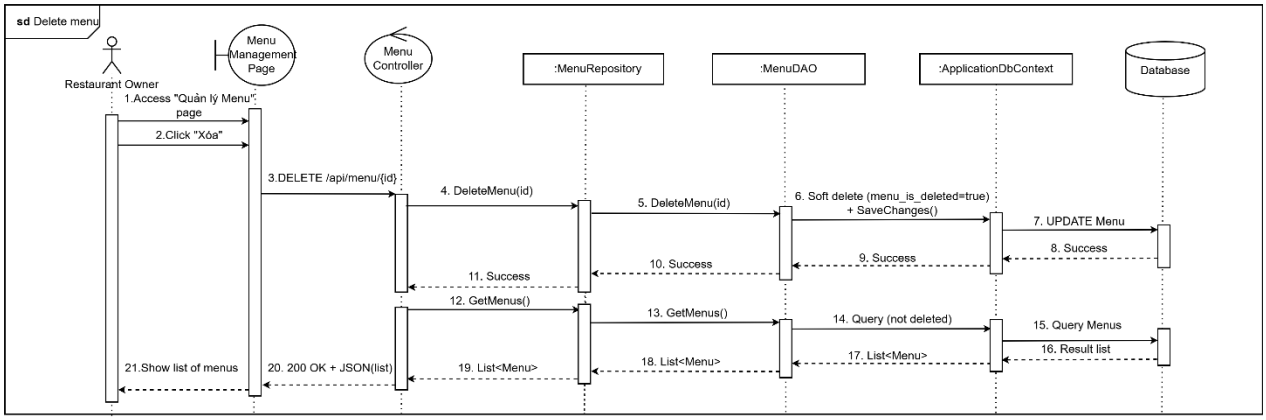


Figure 240: Sequence diagram delete menu (Restaurant Owner)

3.17 Customer

a) Class Diagram

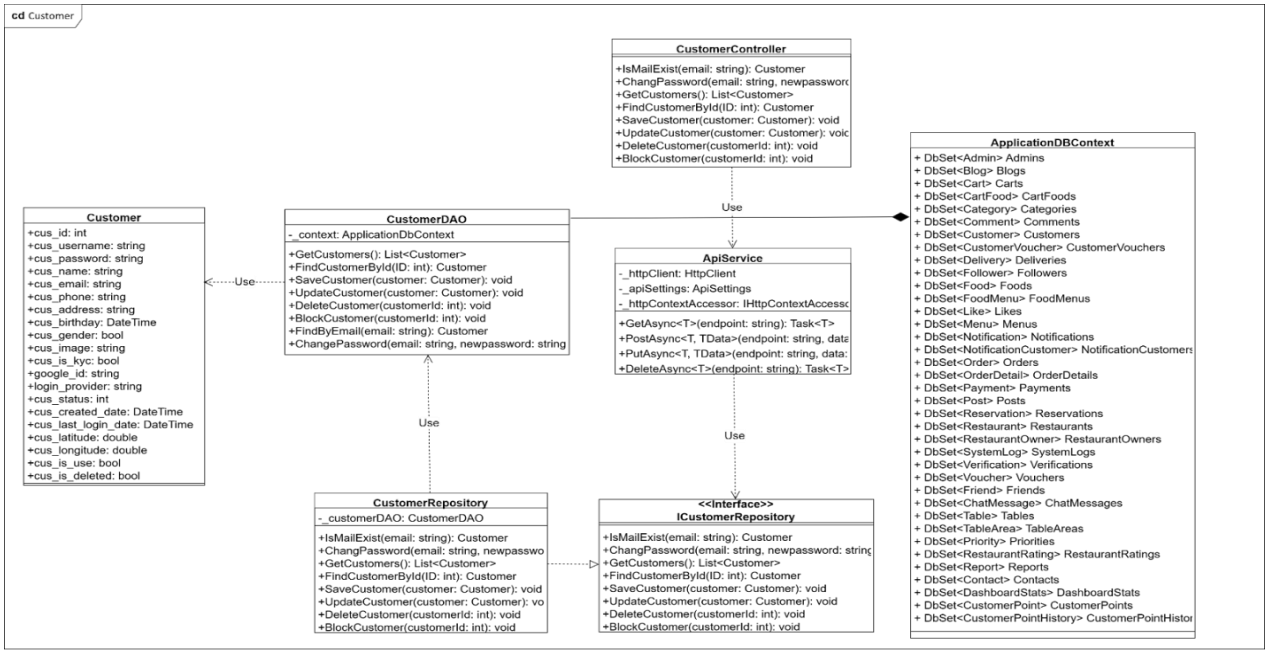


Figure 241: Class diagram of menu

b) Sequence Diagram

View list of customers:

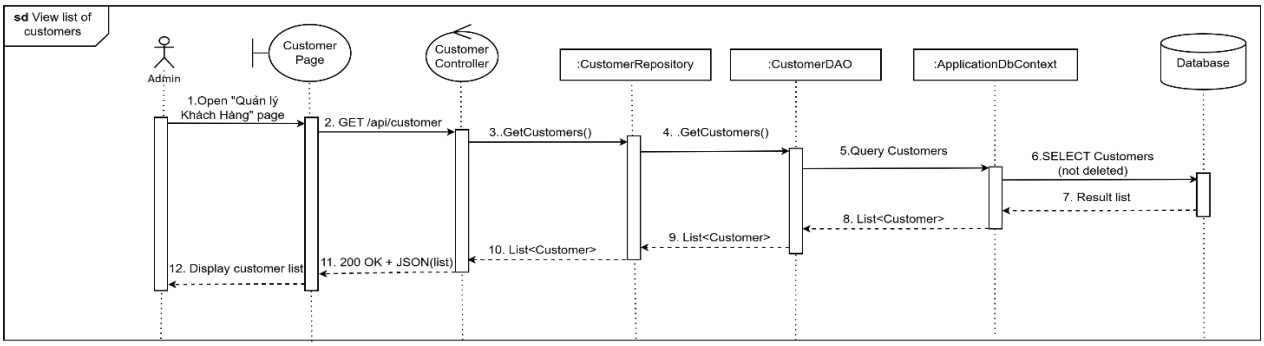


Figure 242: Sequence diagram view list of customers (Admin)

Create customer:

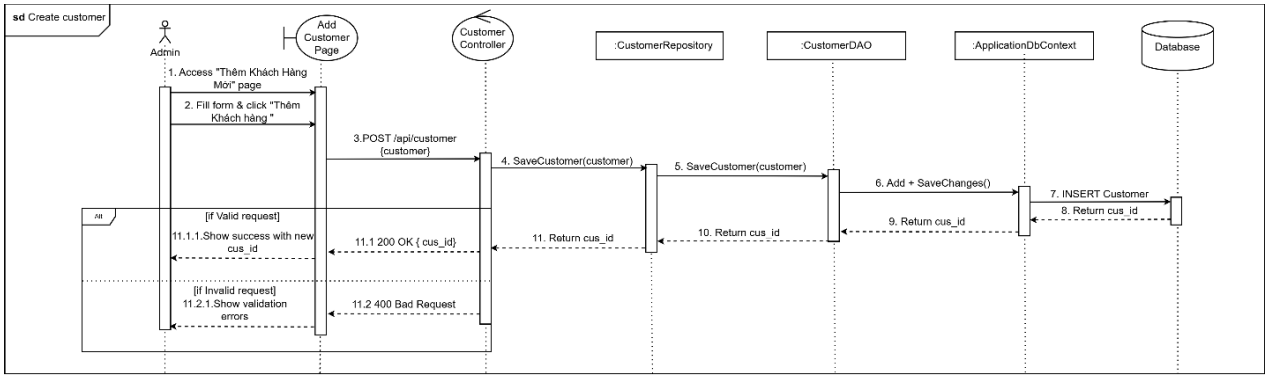


Figure 243: Sequence diagram create customer (Admin)

Delete customer:

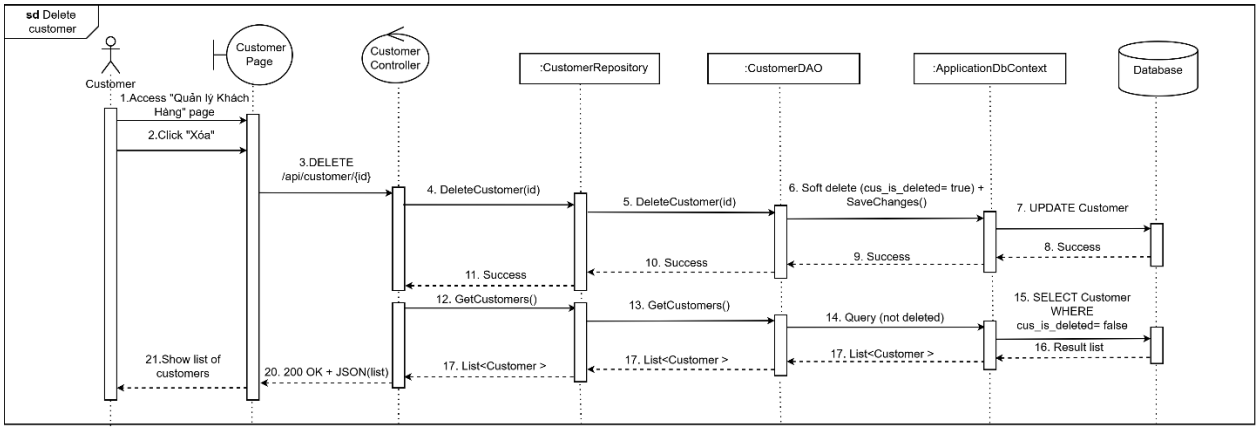


Figure 244: Sequence diagram delete customer (Admin)

Modify customer:

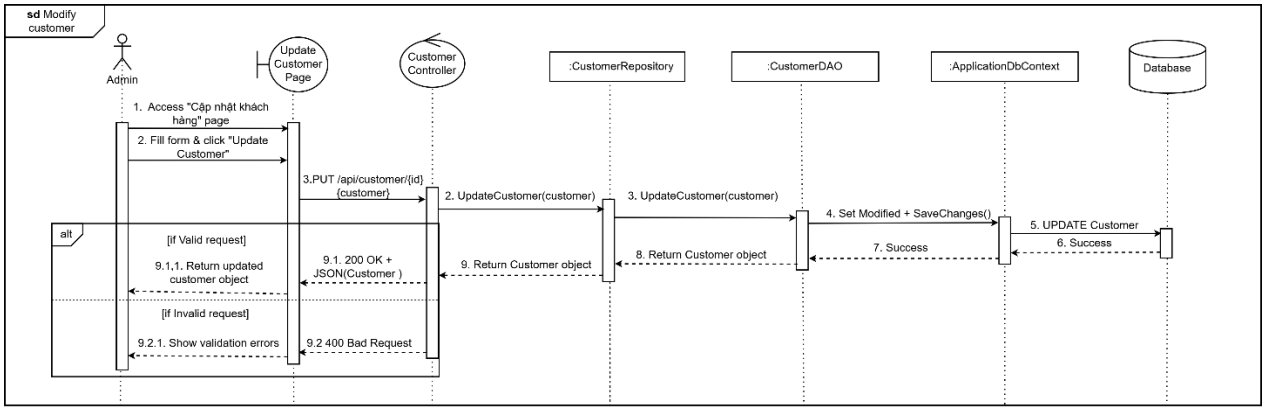


Figure 245: Sequence diagram modify customer (Admin)

Block customer:

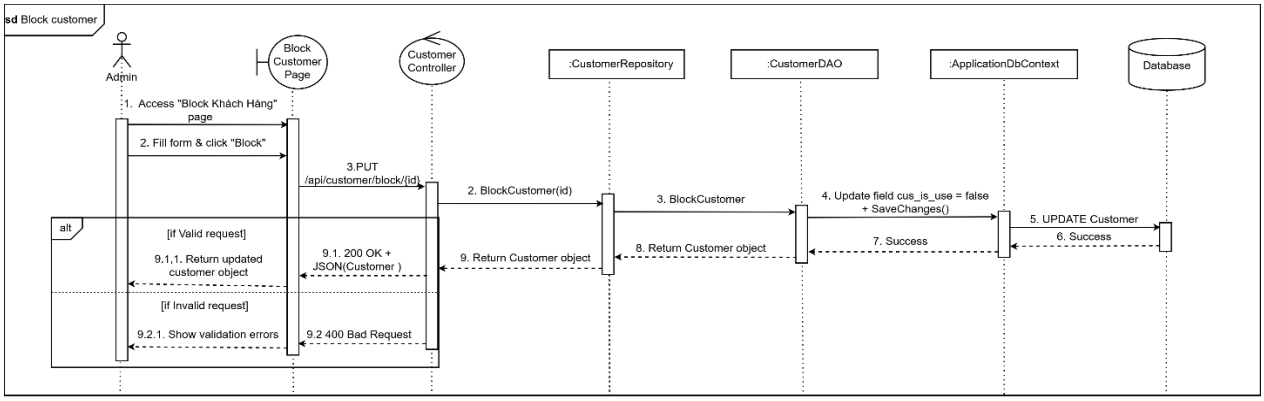


Figure 246: Sequence diagram block customer (Admin)

3.18 Rating

a) Class Diagram

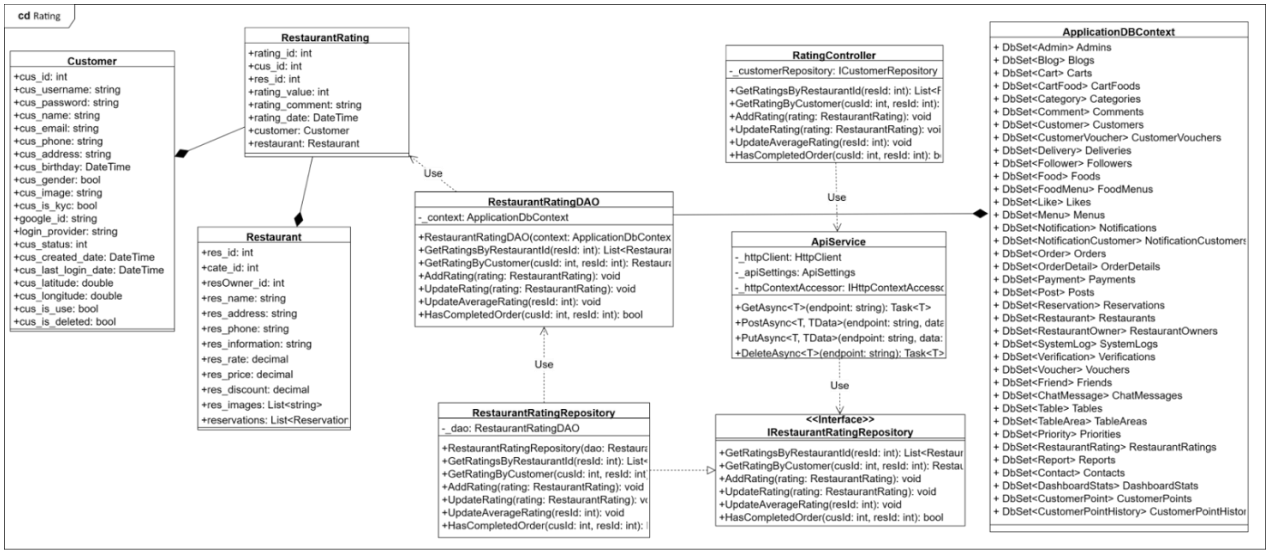


Figure 247: Class diagram of rating

b) Sequence Diagram

Add or update rating:

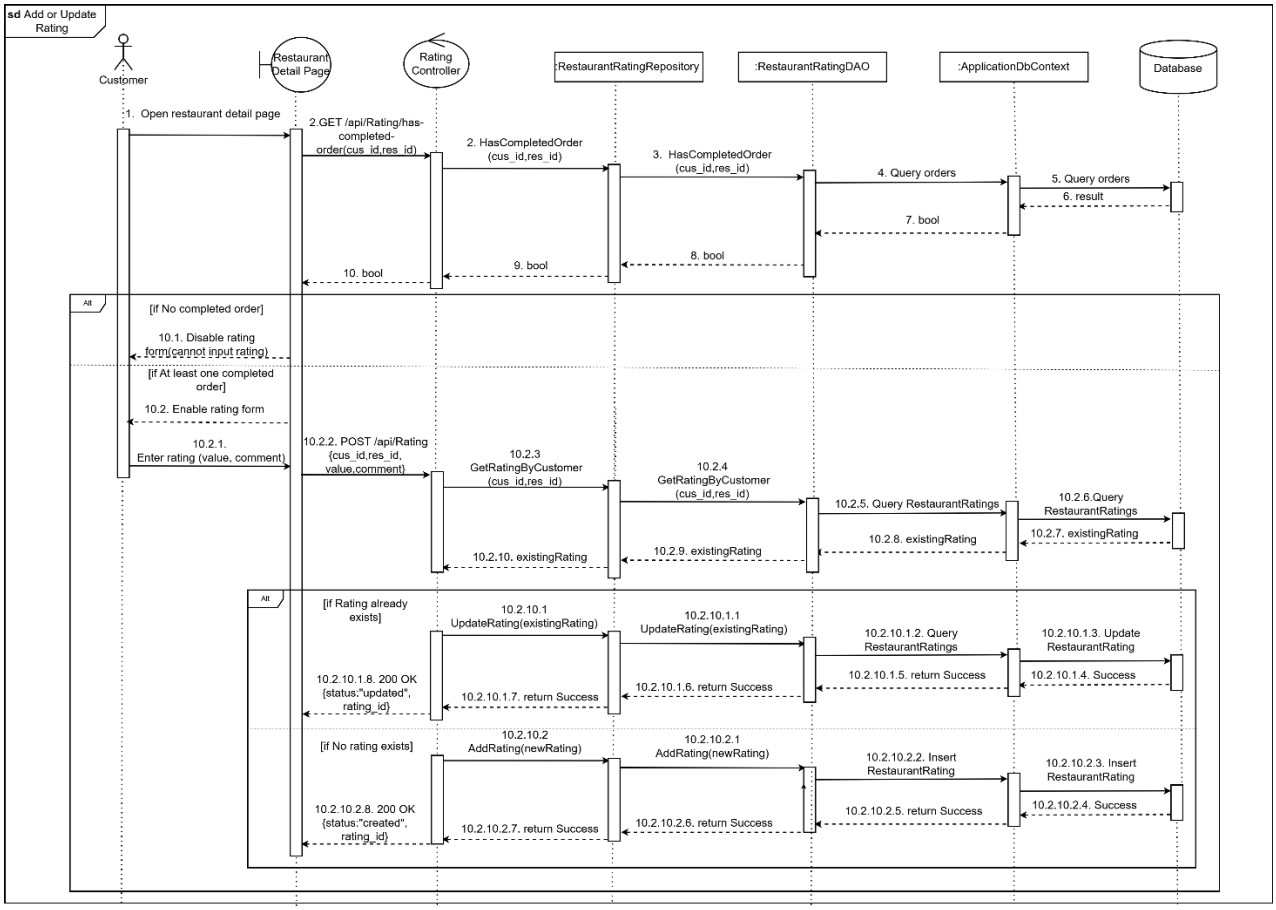


Figure 248: Sequence diagram add or update rating (Customer)

3.19 System Log

a) Class Diagram

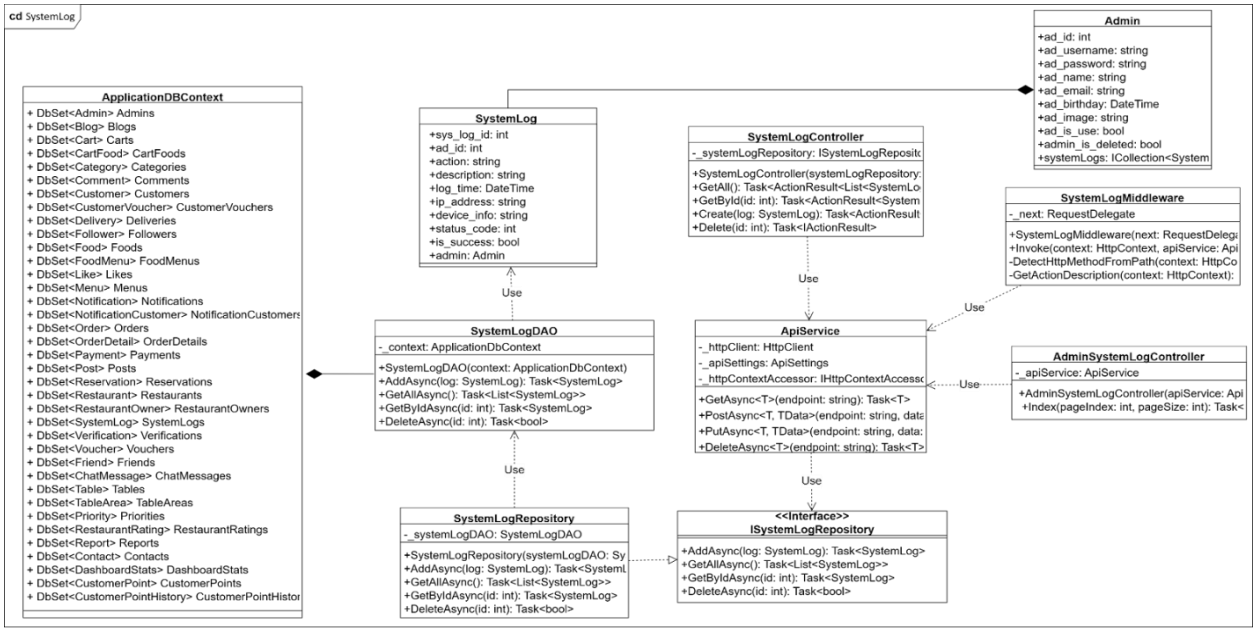


Figure 249: Class diagram of rating

b) Sequence Diagram

View list of logs:

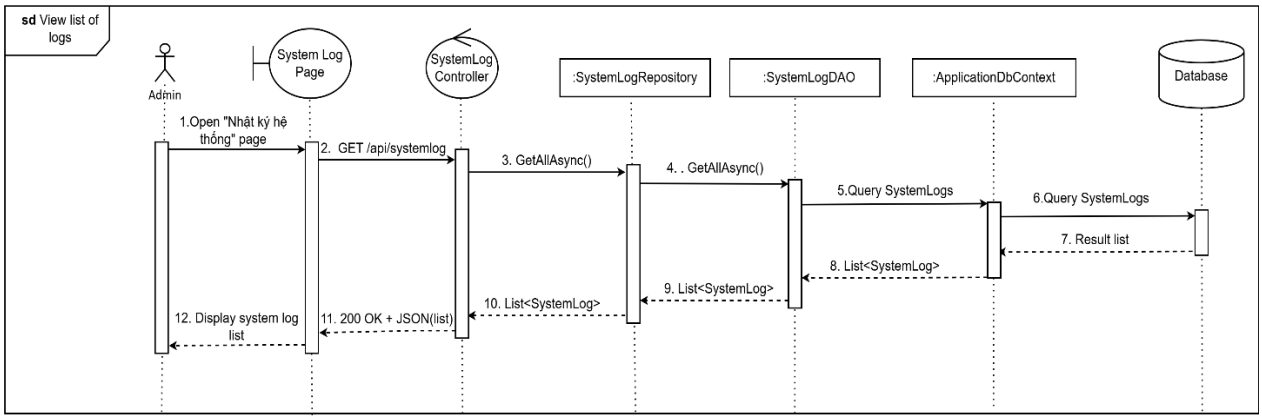


Figure 250: Sequence diagram view list of logs (Admin)

3.20 Chat

a) Class Diagram

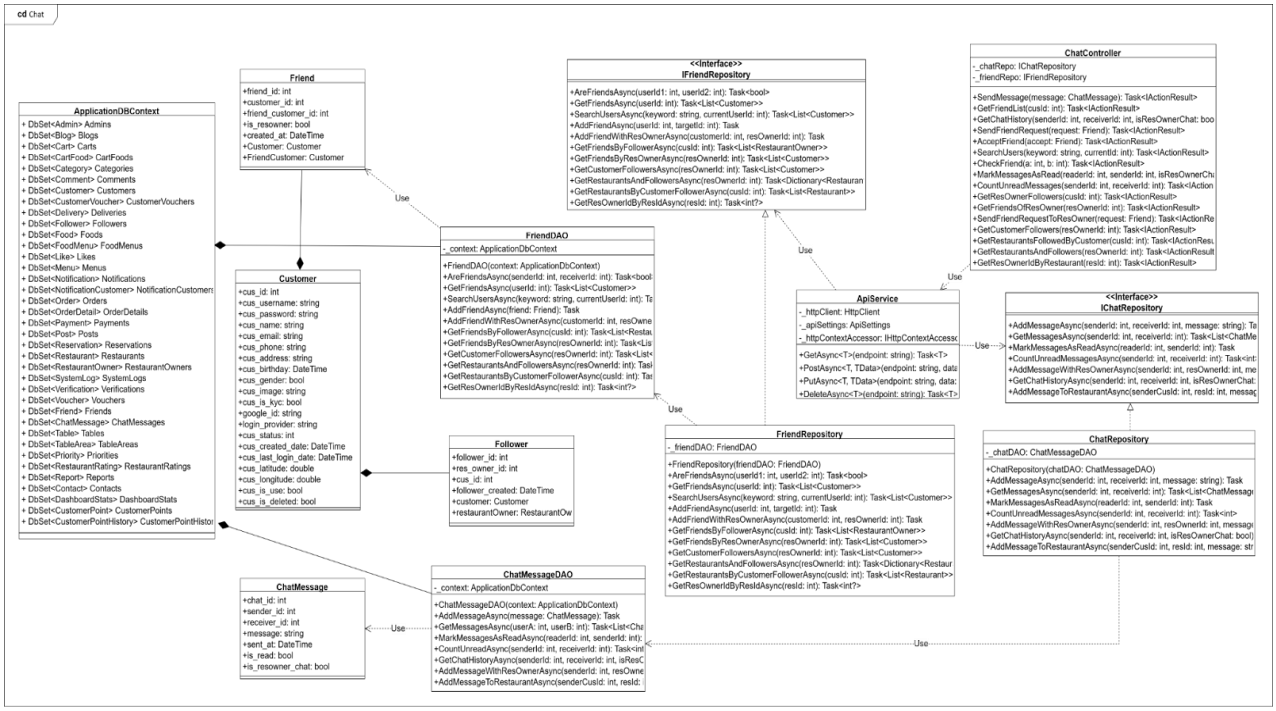


Figure 251: Class diagram of chat

b) Sequence Diagram

Chat between Customers:

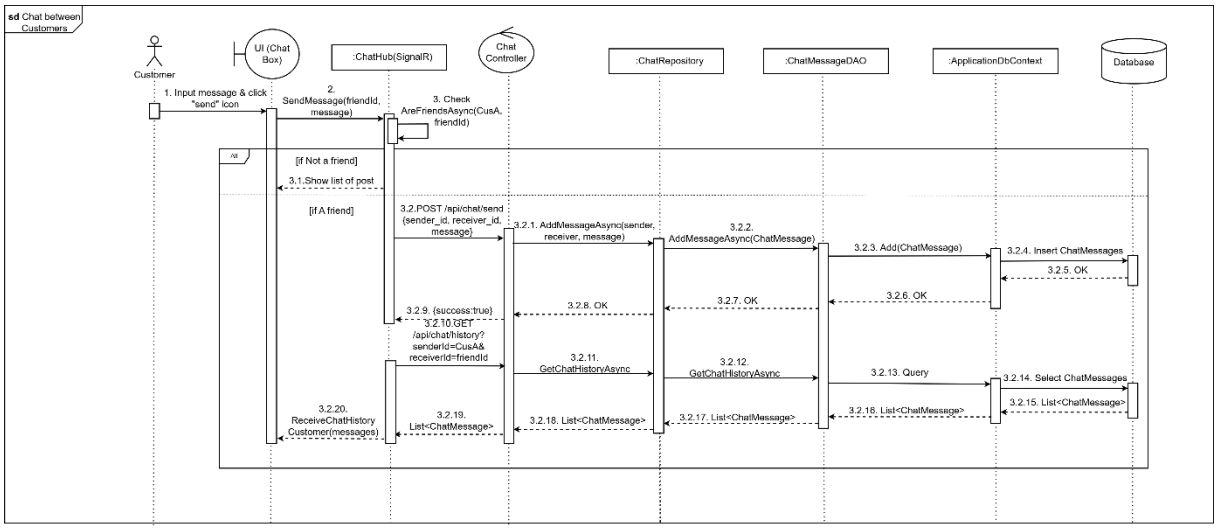


Figure 252: Sequence diagram Chat between Customers (Customer)

V. Software Testing Documentation

1. Scope of Testing

The testing phase of the DineGO system—an online restaurant reservation and food ordering platform—was designed to comprehensively evaluate both functional and non-functional aspects of the system. On the functional side, testing efforts were focused on key modules such as user authentication (including Google login), restaurant and food listing, reservation management, order placement, and payment handling. Administrative features, such as license approval, user management, and content moderation, were also included in the scope to ensure robust performance across all user roles.

In parallel, the team considered non-functional requirements such as performance (e.g., system responsiveness under concurrent access), usability (user-friendly interface and intuitive interactions), and security (authentication and role-based authorization). Given the time constraints and limited access to performance-testing infrastructure, some load testing was kept at a moderate scale. Furthermore, as the system was developed solely for web access, mobile application testing was excluded from the current scope.

2. Test Strategy

To ensure a high level of product quality before deployment, a multi-layered testing strategy was adopted. This included **unit testing**, **integration testing**, **system testing**, and **user acceptance testing**. Each type of testing was selected to serve a distinct purpose and applied using appropriate techniques and tools.

Unit testing was primarily conducted by backend developers to verify the correctness of individual functions and controllers. This phase employed **white-box testing techniques** and utilized the **xUnit** testing framework for automation. Once individual components were verified, integration testing focused on validating the interaction between modules—for example, the communication between controllers, services, and repositories—to detect any mismatches or interface issues.

System testing was executed in a staging environment to simulate full business workflows for both end users and restaurant owners. The objective was to validate the entire system behavior against the specified requirements. Finally, acceptance testing was performed from the client's perspective by walking through realistic user scenarios to ensure the system met their expectations and business goals.

Supporting tools included **Postman** for manual API testing, **Selenium** for automated UI testing, **GitHub Issues** for tracking bugs and fixes, and **xUnit** for unit testing. These tools were selected based on their reliability, community support, and alignment team's technical stack.

2.1 Testing Types

To ensure comprehensive evaluation of the system's reliability and performance, the project applies several testing types throughout the development cycle.

Functional testing serves as the foundation, aiming to validate that each feature behaves according to the specified requirements. This is carried out primarily through black-box techniques, where the testers verify outputs based on expected results without considering internal code logic.

Unit testing complements this by targeting individual functions or methods using white-box techniques. It allows developers to identify logical errors early in development and helps improve code maintainability.

Integration testing is used to verify interactions between modules, such as between the authentication controller and the user service or between the reservation API and the database layer. This ensures that the system modules cooperate correctly when combined.

Finally, **acceptance testing** is performed from the end-user's perspective. It is used to validate the overall business flow and whether the system meets client expectations under real-world scenarios.

2.2 Test Levels

To ensure thorough quality control throughout the development lifecycle, the testing process in this project is categorized into multiple levels. Each level focuses on a distinct layer of the system and is associated with specific testing objectives. Unit testing serves as the foundation by verifying the correctness of individual components such as functions, methods, and classes. Integration testing follows, which ensures that modules interact correctly once they are connected. System testing is conducted on the entire software product to validate that it meets both functional and non-functional requirements. Finally, acceptance testing is performed from the user's perspective to confirm whether the system delivers the expected outcomes and business value in real-world conditions.
The table below illustrates the mapping between test types and the corresponding test levels applied in this project:

Table 20: Test level table

Type of Tests	Test Level			
	Unit	Integration	System	Acceptance
Functional Testing	X		X	X
User Interface Testing			X	X
Performance Testing			X	

2.3 Supporting Tools

The use of supporting tools plays a crucial role in maintaining consistency, improving test accuracy, and automating repetitive testing tasks. For this project, we selected tools that align well with the technology stack of the system. Each tool was carefully chosen based on its ease of use, community support, integration capabilities, and effectiveness in detecting errors across different testing levels. The table below provides detail about the tools employed in this project:

Table 21: Supporting tools table

Purpose	Tool	Vendor/In-house	Version
Unit Testing	xUnit	Open-source	Latest
API Testing	Postman	Open-source	v10+
UI Testing	Selenium	Open-source	v4
Bug Tracking	GitHub Issues	In-house	N/A

3. Test Plan

3.1 Human Resources

Testing responsibilities are distributed among team members according to their areas of expertise. Each member is assigned a well-defined role to ensure thorough and coordinated testing coverage across all system modules. The QA lead is responsible for defining the overall test strategy and verifying results. Backend and frontend testers focus on validating functional and UI aspects respectively, while

integration QA handles scenarios spanning multiple layers. The test support role ensures data consistency and environment stability. The table below summarizes the roles and responsibilities:

Table 22: Human resources table

Worker/Doer	Role	Specific Responsibilities/Comments
Phuonghh	QA Lead	Coordinate testing activities, design test cases, validate test results
Sieuhhd	Backend Tester	Conduct unit testing for APIs and services
Khoinv	Frontend Tester	Execute UI validation and layout behavior testing
Thangtm	Integration QA	Perform integration testing across modules
Thanhdt	Test Support	Prepare testing data and set up testing environment

3.2 Test Environment

A stable and replicable test environment is essential to ensure the reliability of test results. The project employs a mixture of development and cloud-based tools that simulate production-like conditions. APIs are deployed in a staging environment for system testing, while front-end tests are executed across different devices and browsers to verify compatibility. The following table outlines the environment configuration:

Table 23: Test environment table

Purpose	Tool	Provider	Version
Write and run unit tests	Visual Studio Code	Microsoft	Latest
Deploy and test backend APIs	ASP.NET Core	In-house	Latest
Prepare seed data	SQL Server	Microsoft	2019+
Execute UI tests	Selenium	Selenium	Latest

3.3 Test Milestones

The testing process is divided into specific milestones that mark the completion of critical testing activities. Each milestone is aligned with development phases to enable early error detection and prevent defect accumulation. These checkpoints also help manage time effectively and prepare for final product acceptance. The milestone schedule is illustrated below:

Table 24: Test milestones table

Milestone Task	Start Date	End Date
Complete unit testing of login, reservation modules	2025-07-10	2025-07-14
Finish integration testing for all service workflows	2025-07-15	2025-07-19
Deploy and execute full system testing	2025-07-20	2025-07-23
Conduct acceptance testing by the client	2025-07-24	2025-07-26

4. Test Cases

To ensure that all system functionalities operate correctly and according to the requirements, a detailed set of test cases was prepared. These test cases cover both unit-level validations for individual methods and higher-level system workflows involving multiple components.

Each test case includes clearly defined objectives, preconditions, input data, expected results, and actual results. The testing team applied both **positive** and **negative** testing strategies to detect unexpected behavior or system failures under invalid input.
For organizational clarity, all test cases were divided into two main documents:

- **Unit Test Cases**: Defined and executed for core business logic, particularly within the service and controller layers. [4]
- **Other Test Cases**: Covering Integration Testing (IT), System Testing (ST), and Acceptance Testing (AT), executed according to real-user scenarios. [5]

The test cases were reviewed regularly to ensure alignment with newly updated requirements and implemented features.

5. Test Reports

The test results are compiled into structured reports, reflecting the effectiveness of the testing strategy. The reports include key indicators such as:

- Number of test cases executed
- Number of test cases passed/failed
- Defect types and their severity
- Bug resolution time and test cycle coverage

After each testing cycle, the QA team performed an analysis to identify defect-prone modules and adjusted the test focus accordingly. Overall, the test reports demonstrated high success rates in the unit and system test phases, indicating stability of the core features. Minor issues encountered during acceptance testing were addressed promptly and validated through regression tests.
The detail of the test execution result, and defect logs can be found in the accompanying test report files, especially. [5]

5.1 Unit Test Case

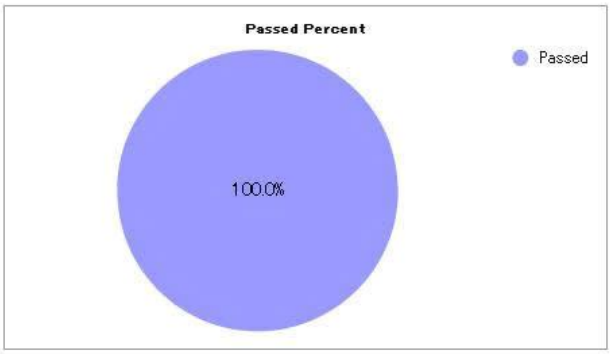
UNIT TEST REPORT

Project Name	*Smart Restaurant Reservation System*	**Creator**	Duong Tuan Thanh
Project Code	*SRRS*	**Reviewer/Approver**	Ngo Giang Thanh
Document Code	*SRRS_Test Report_v1.0*	**Issue Date**	*2/7/2025*
Notes			

No	Function code	Passed	Failed	Untested	N	A	B	Total Test Cases
1	Login	9	0	0	2	7	0	9
2	Register	9	0	0	2	6	1	9
3	ForgotPassword	4	0	0	1	3	0	4
4	UpdateProfile	7	0	0	2	3	2	7
5	ChangePassword	7	0	0	1	5	1	7
6	CreateReservation	7	0	0	1	4	2	7
7	UpdateReservation	6	0	0	1	4	1	6
8	DeleteReservation	4	0	0	1	3	0	4
9	GetAllReservation	5	0	0	1	4	0	5
10	FindReservationByID	4	0	0	1	3	0	4
11	AddRestaurant	7	0	0	2	4	1	7
12	UpdateRestaurant	8	0	0	3	4	1	8
13	DeleteRestaurant	4	0	0	1	3	0	4
14	GetAllRestaurant	5	0	0	1	4	0	5
15	FindRestaurantByID	4	0	0	1	3	0	4
16	GetOrder	4	0	0	1	3	0	4
17	GetOrderById	5	0	0	1	4	0	5
18	UpdateOrder	4	0	0	1	3	0	4
19	AddToCart	6	0	0	2	4	0	6
20	RemoveFromCart	5	0	0	1	4	0	5
21	UpdateCartItem	6	0	0	1	5	0	6
22	AddFood	7	0	0	1	5	1	7
23	DeleteFood	4	0	0	1	3	0	4
24	UpdateFood	7	0	0	1	5	1	7
25	GetAllFoods	5	0	0	1	4	0	5
26	AddMenu	9	0	0	2	7	0	9
27	UpdateMenu	10	0	0	3	7	0	10
28	DeleteMenu	4	0	0	1	3	0	4
29	GetAllMenus	5	0	0	1	4	0	5
30	CreateBlog	11	0	0	2	9	0	11
31	UpdateBlog	11	0	0	3	8	0	11
32	DeleteBlog	5	0	0	5	0	0	5
33	GetAllBlog	7	0	0	2	5	0	7
34	CreatePost	6	0	0	1	3	2	6
35	UpdatePost	8	0	0	2	5	1	8
36	GetPosts	5	0	0	1	4	0	5
37	AddComment	7	0	0	1	5	1	7
38	DeleteComment	4	0	0	1	3	0	4
39	GetCommentById	5	0	0	1	4	0	5
40	SendMessage	5	0	0	1	4	0	5
41	GetFriendLists	4	0	0	2	2	0	4
42	GetChatHistory	15	0	0	4	7	4	15
43	SendFriendRequest	15	0	0	3	7	5	15
44	SearchUsers	14	0	0	6	6	2	14
45	GetRestaurantsAndFollowers	10	0	0	4	4	2	10
46	SuggestFood	14	0	0	7	5	2	14
47	SetManualWeight	17	0	0	6	6	5	17
48	UpdatePriority	17	0	0	8	7	2	17
49	getAllLikes	5	0	0	2	3	0	5
50	ReactToPost	16	0	0	4	12	0	16
51	GetReactionsByPost	8	0	0	2	6	0	8
52	GetAllVouchers	5	0	0	1	4	0	5
53	GetVoucherById	5	0	0	1	4	0	5
54	GetVoucherByCode	6	0	0	1	4	1	6
	Sub total	396	0	0	110	248	38	396

Figure 253: Unit Test case UC List

Test coverage	**100.00 %**
Test successful coverage	**100.00 %**
Normal case	**27.78 %**
Abnormal case	**62.63 %**
Boundary case	**9.60 %**



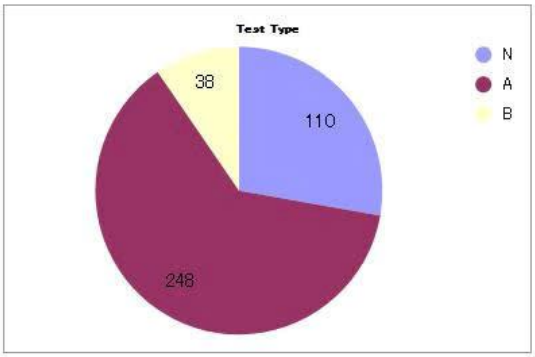


Figure 254: Unit Test Case percentage chart

5.2 System Test

TEST STATISTICS

Project Name	*Smart Restaurant Reservation System*	**Creator**		Duong Tuan Thanh
Project Code	*SRRS*	**Reviewer/Approver**		Ngo Giang Thanh
Document Code	*SRRS_Test Report_v1.0*	**Issue Date**		
Notes				

No	Module code	Passed	Failed	Pending	N/A	Number of test cases
1	Login	15	0	0	0	15
2	Register	15	0	0	0	15
3	Customer	29	0	0	0	29
4	Restaurant	21	0	0	0	21
5	Food	19	0	0	0	19
6	Menu	18	0	0	0	18
7	Cart	14	0	0	0	14
8	Order	6	0	0	0	6
9	Reservation	10	0	0	0	10
10	Blog	26	0	0	0	26
11	Post	27	0	0	0	27
12	Friend	14	0	0	0	14
13	Chat	20	0	0	0	20
14	Voucher	21	0	0	0	21
15	Notification	15	0	0	0	15
16	SystemLog	6	0	0	0	6
	Sub total	276	0	0	0	276

Test coverage	**100.00 %**
Test successful coverage	**100.00 %**

Figure 255: System test case report

5.3 Test Conclusion

Comprehensive testing was performed on all modules, covering normal, error, and boundary scenarios. All planned cases passed, showing the system is stable and ready for release

- **Results:** 100% coverage, 100% pass, no failures.
- **Scope:** 16 modules, 276 system tests, 396 unit tests.
- **Test types:** Normal 27.78%, Abnormal 62.63%, Boundary 9.60%.

VI. Release Package & User Guides

1. Deliverable Package

This table lists all deliverables of the DineGO project, including code packages, database scripts, and documentation. Each deliverable is organized by category, with corresponding sub-items and version numbers for reference:

Table 25: Deliverable Package

No.	Items	Sub-items	Version
Code Package			
1	DineGO_Api	DineGO Api	1.0
2	DineGO_Admin	DineGO Admin Web	1.0
3	DineGO_Client	DineGO Client Web	1.0
4	DineGO_Api/AI_Prediction	DineGO Machine Learning	1.0
Database			
1	Tables and data	DineGo_DB.sql	1.0
Documents			
1	Report 1	Report1_Project Introduction	1.0
2	Report 2	Report2_Project Management Plan	1.0
3	Report 3	Report3_Software Requirement Specification	1.0
4	Report 4	Report4_Software Design Document	1.0
5	Report 5	Report5_Test Documentation	1.0
		Report5_Test Case Document	1.0
		Report5_Unit Test Case	1.0
6	Report 6	Report6_Software User Guides	1.0
7	Report 7	Report7_Final Project Report	1.0

2. Installation Guides

2.1 System Requirements

The following are the minimum system requirements for deploying and running the DineGO system, including its modules such as DineGO_Api, DineGO_Admin, DineGO_Client, and DineGO_Prediction:

Server Requirements:

- Operating System: Windows Server 2022 or higher.
- Processor: For development/testing: Intel® Core™ i7/i9 (10th Gen or higher, minimum 4 cores / 8 threads).
- RAM: Minimum 8 GB.
- Storage: Minimum 60 GB SSD available.
- Database: Microsoft SQL Server 2022.
- Network: Stable internet connection; ports for HTTP/HTTPS and SQL Server must be open.
- Additional Software:
    - .NET 5.0 SDK or higher.
    - IIS (Internet Information Services) for web application hosting.
    - Python 3.10+ (required for DineGO_Prediction module).
    - Python packages: pandas, scikit-learn, flask, etc.

Client Requirements:

- **Operating system:** Windows 7 or higher
- **Web Browser**: Latest version of Chrome, Edge, or Firefox.
- **JavaScript**: Must be enabled.

- **Cookies**: Enabled for session handling.

2.2 Setup file

- DineGO.zip (including DineGo_Api, DineGO_Client, DineGO_Admin, DineGO_Prediction)
- Dotnet-hosting-7.0.18-win.exe
- SQL2022-SSEI-Dev.exe
- SSMS-Setup-ENU.exe
- DineGo_DB.sql

2.3 Installation Instruction

a. Install SQL Server 2022 and SSMS 20

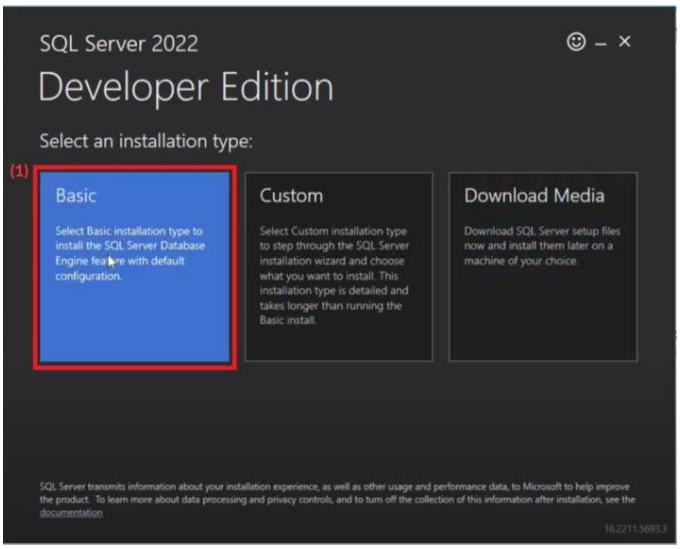


Figure 256: Install SQL Server 2022 and SSMS 2019

- (1) Run the SQL2022-SSEI-Dev.exe installation file and select “Basic”

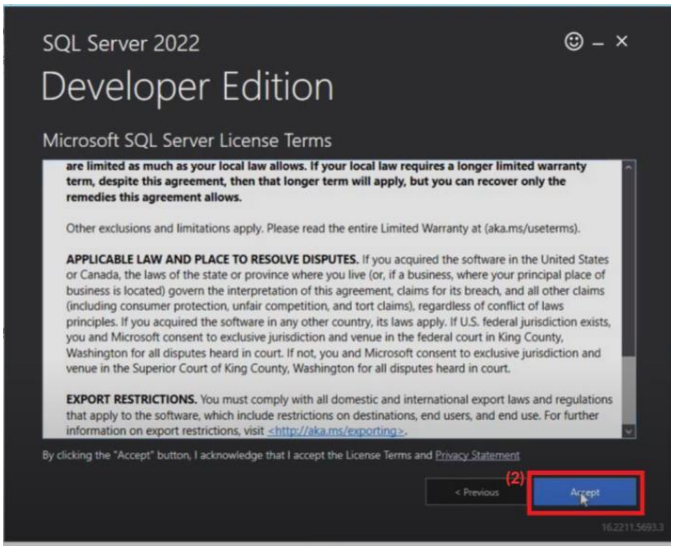


Figure 257: Accepting the license terms

- (2) Scroll down and select “Accept”

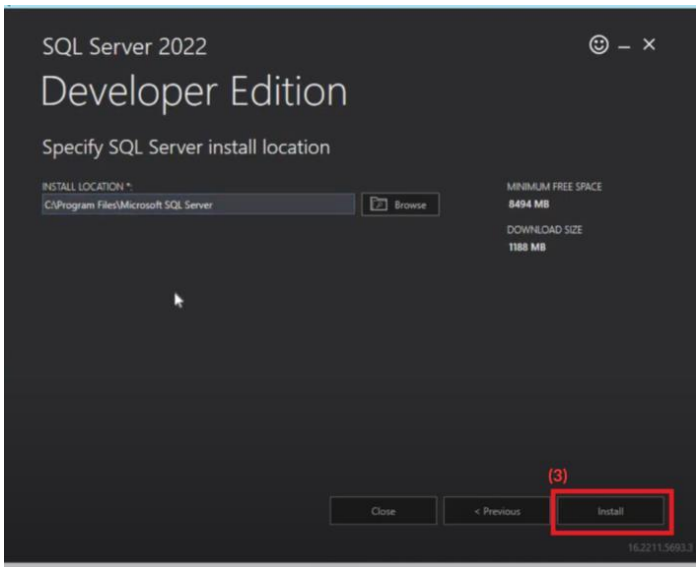


Figure 258: Specifying installation location and starting installation

- (3) Select “Install”

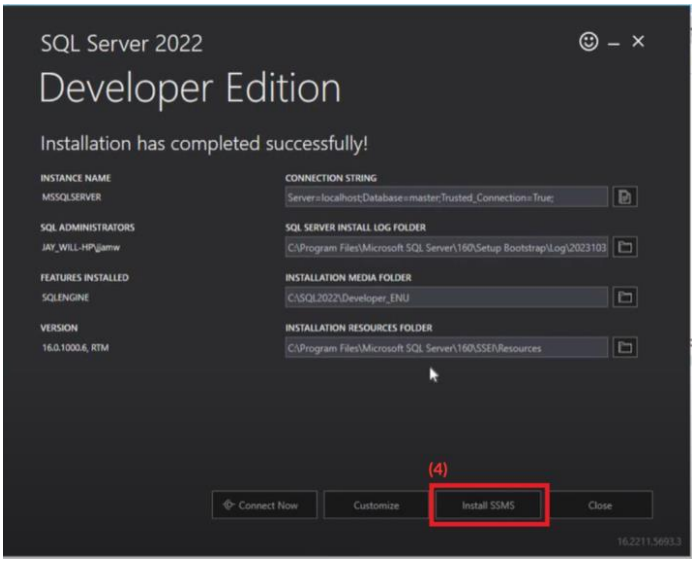


Figure 259: installation completed with option to install SSMS

- (4) Select “Install SSMS” and run the file SSMS-Setup-ENU.exe

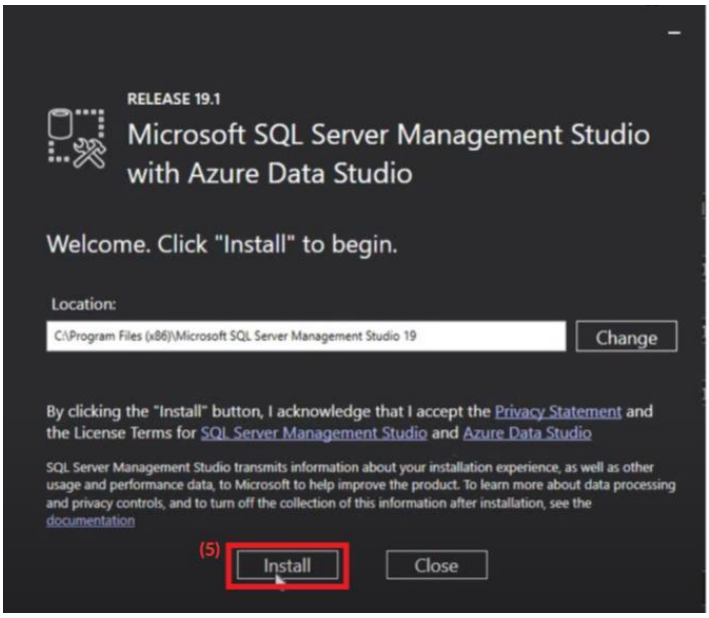


Figure 260: Selecting location and starting installation

- (5) Select “Install

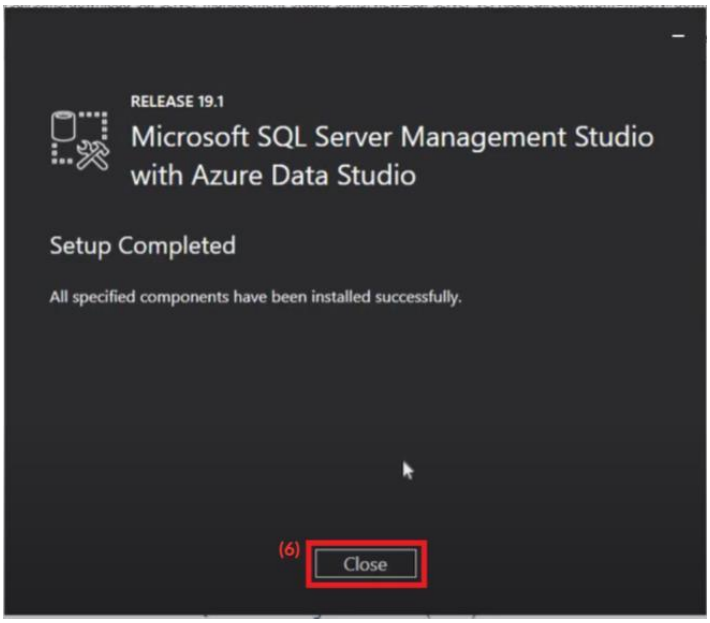


Figure 261: Installation completed successfully

- (6) Select “Close”

b. Run file backup database

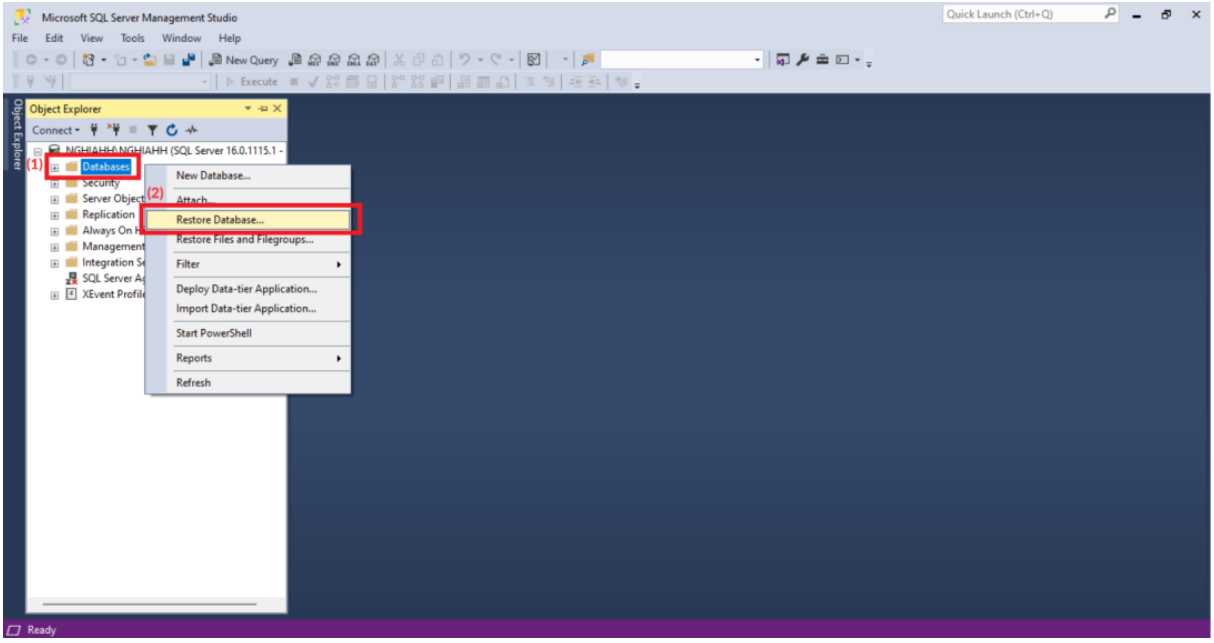


Figure 262: Backup database

- (1) Right-click and select “Database”
- (2) Select “Restore Database”

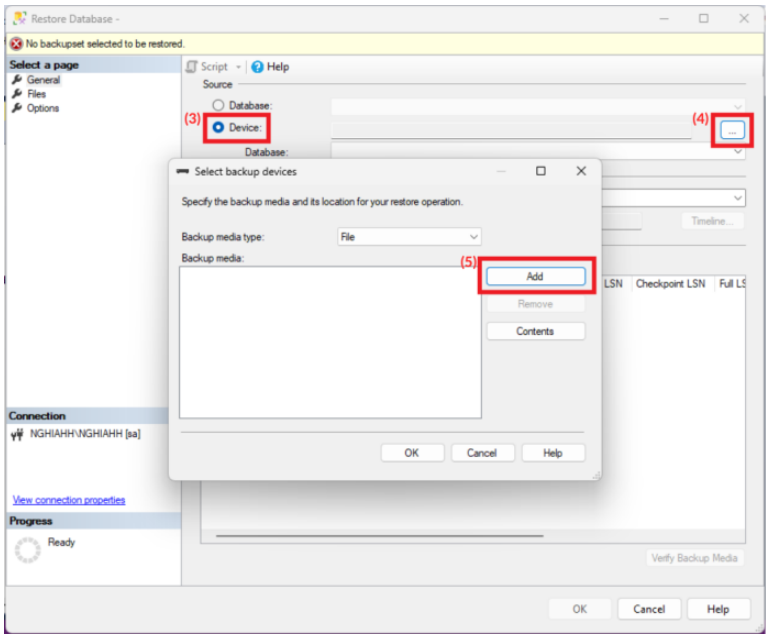


Figure 263: Selecting device and adding backup

- (3) Select “Device”
- (4) Select “...”

- (5) Select “Add”

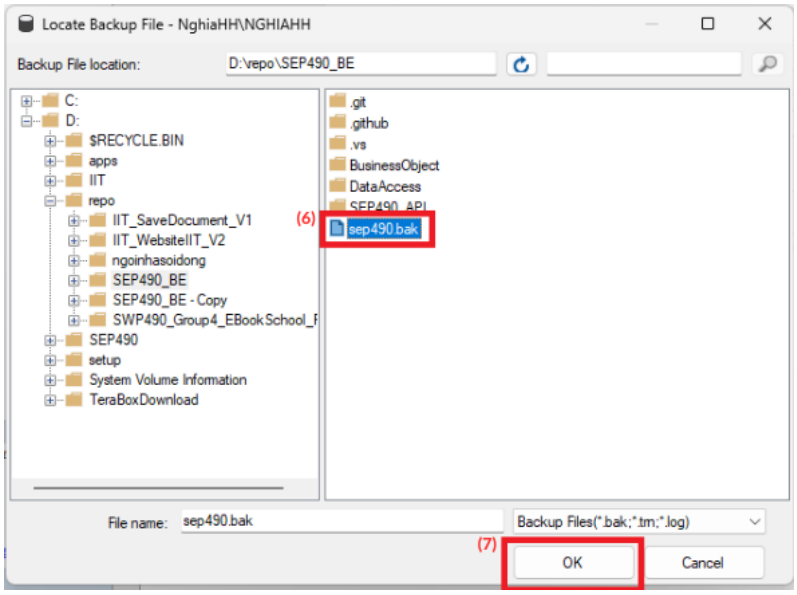


Figure 264: Selecting the backup file (.bak)

- (6) Point to and select the sep490.bak file
- (7) Select “OK”

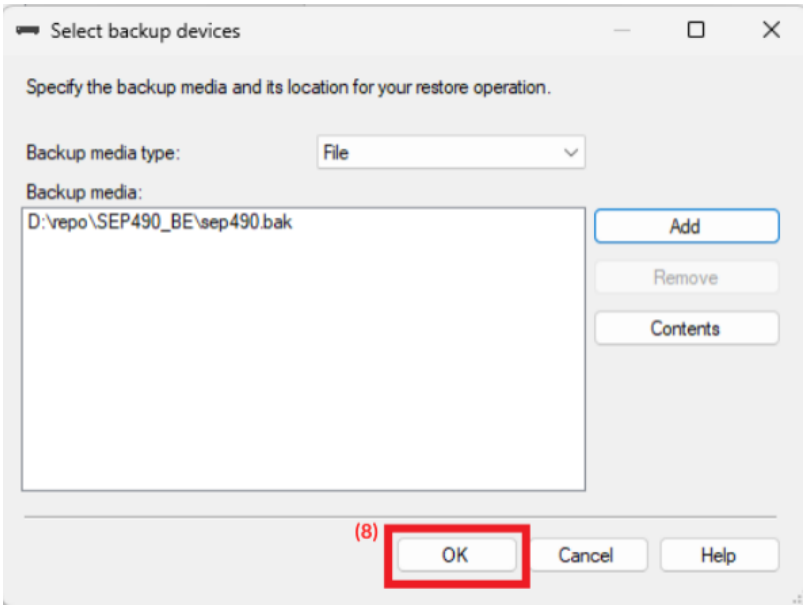


Figure 265: Confirming the selected backup

- (8) Select “OK”

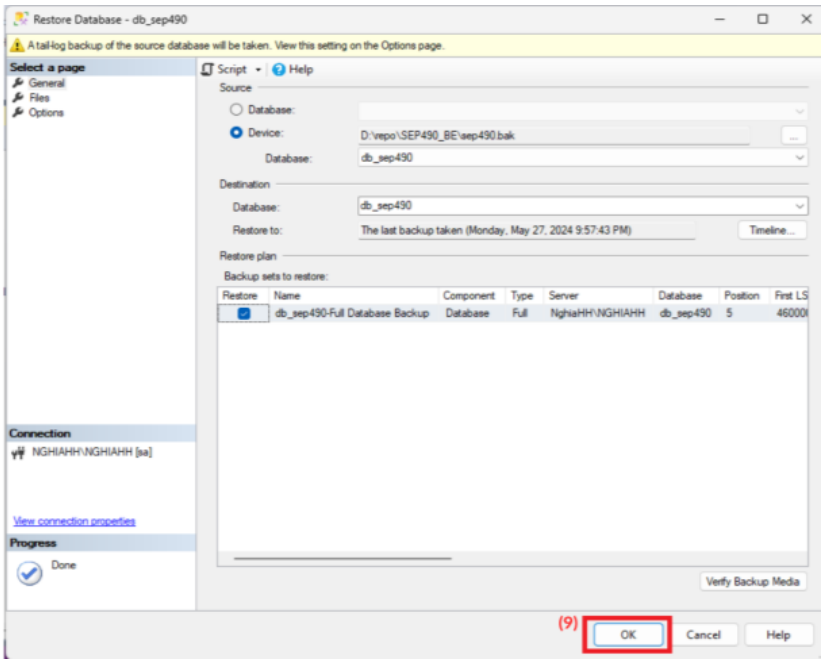


Figure 266: Final confirmation

- (9) Select "OK"

c. Install .NET Hosting Bundle 5.0.17

- Run the file dotnet-hosting-5.0.17-win.exe

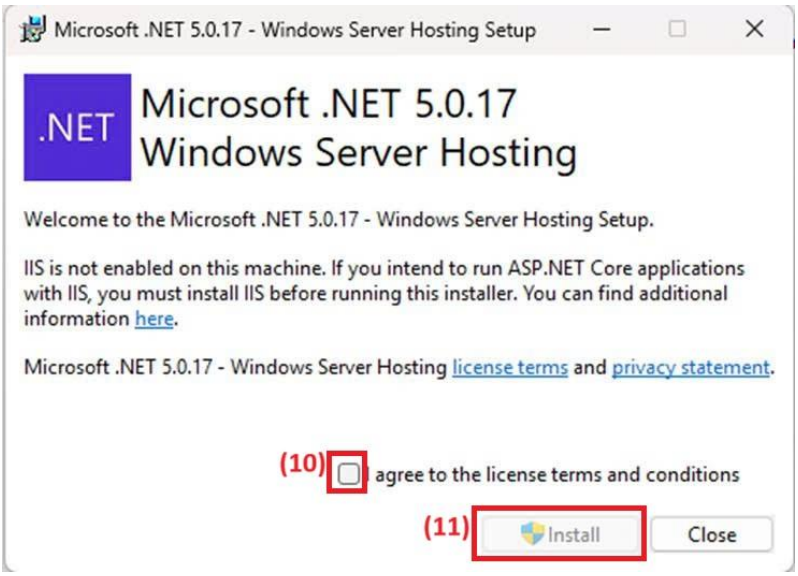


Figure 267: Install .NET Hosting Bundle 5.0.17

- (10) Check the "Agree to the license terms and conditions" button
- (11) Select "Install"

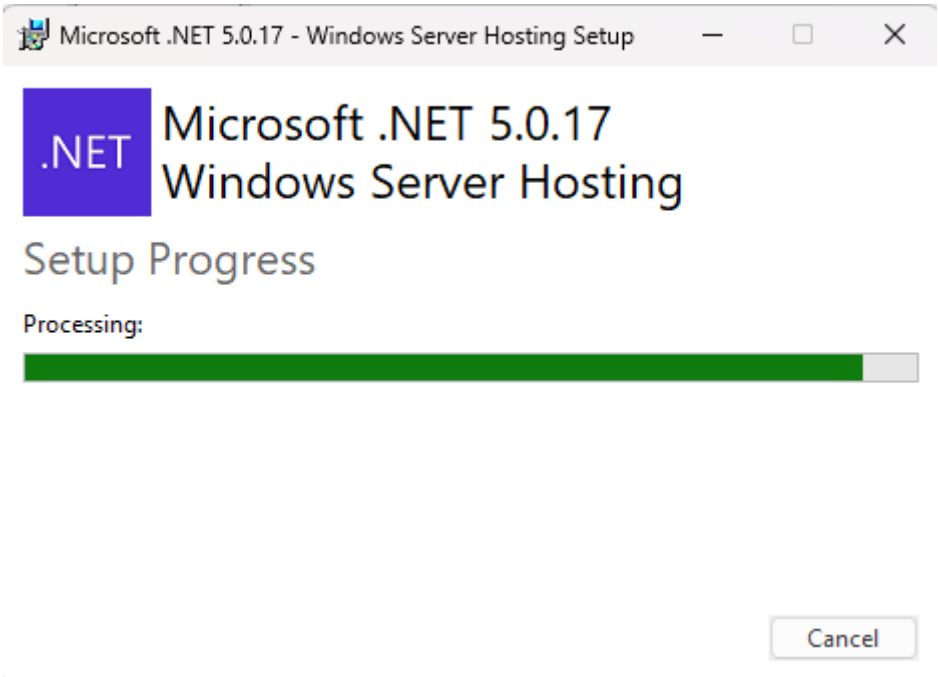


Figure 268: Wait install process

- Wait for the installation process

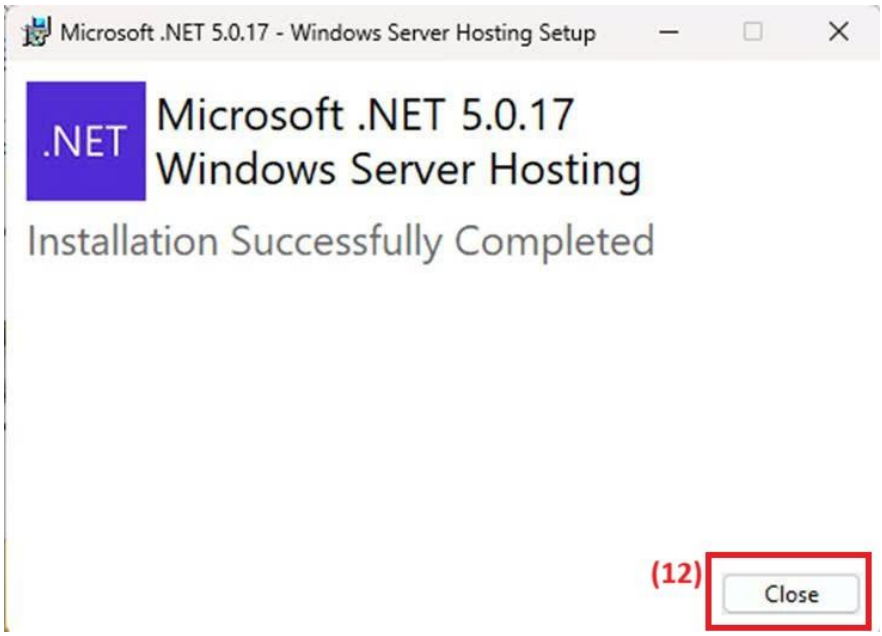


Figure 269: Install complete

- (12) Select “Close”

d. Install IIS Manager

● Open Server Manager and select Dashboard

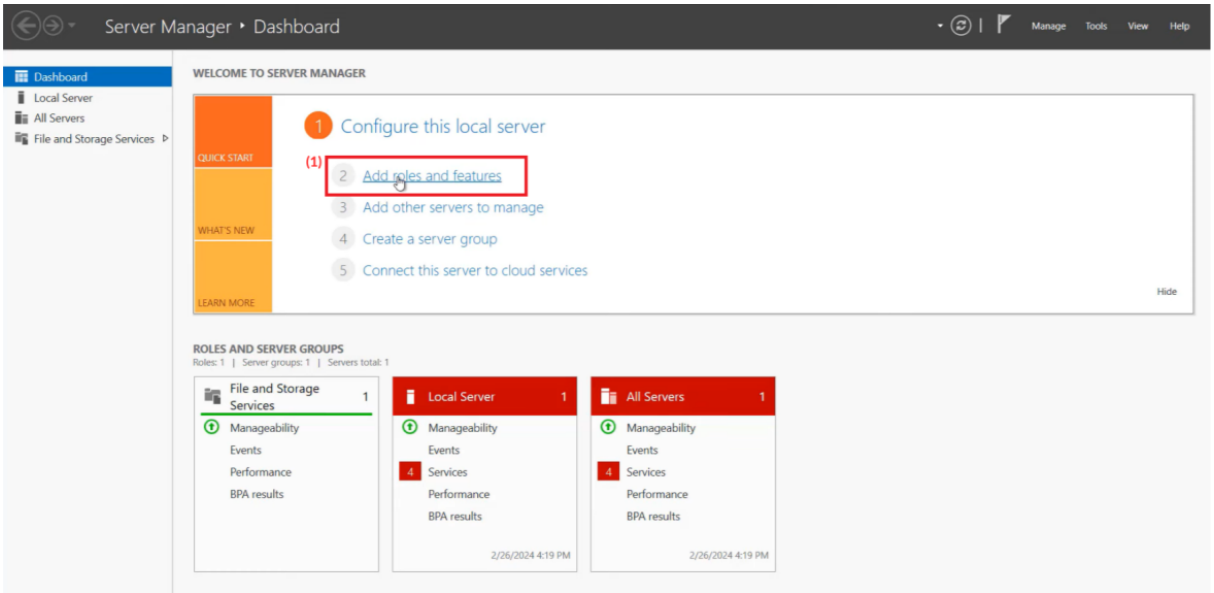


Figure 270: Install IIS Manager

● (1) Select “Add roles and features”

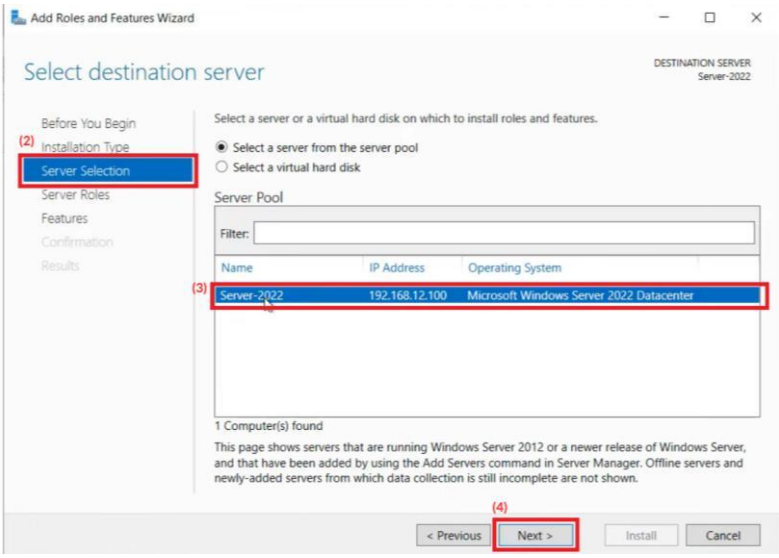


Figure 271: Selecting the destination server

● (2) Select “Next” to “Server Selection”

● (3) Select “Server 2022”

● (4) Select “Next”

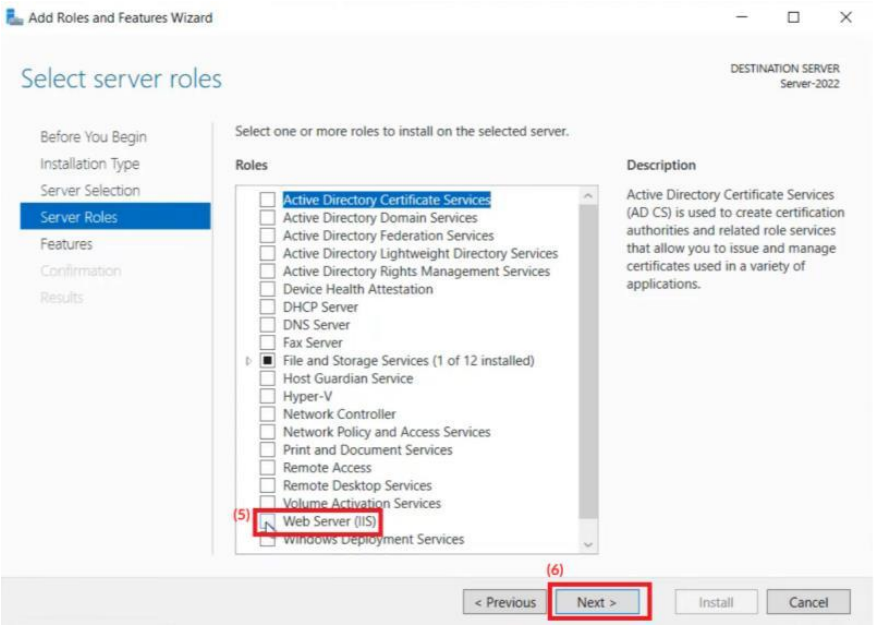


Figure 272: Server role selection in Windows Server 2022

- (5) Click “Web Server (IIS)”
- (6) Select “Next”

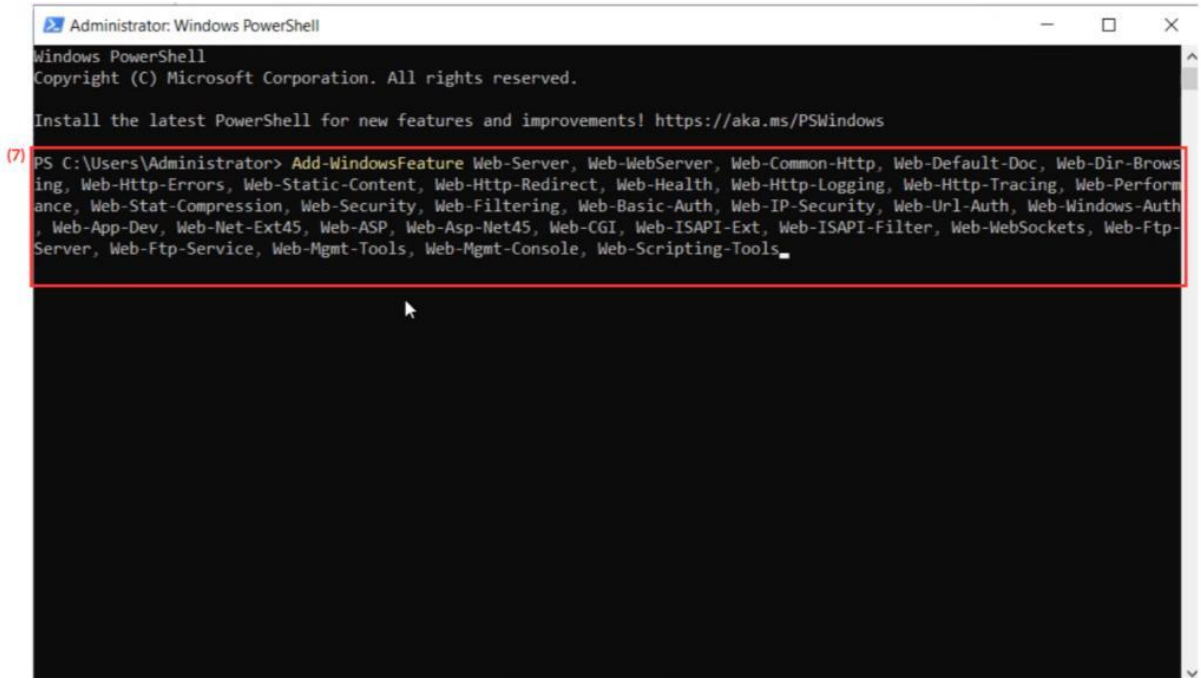


Figure 273: PowerShell command for installing Web Server (IIS)

- (7) Open terminal and check feature

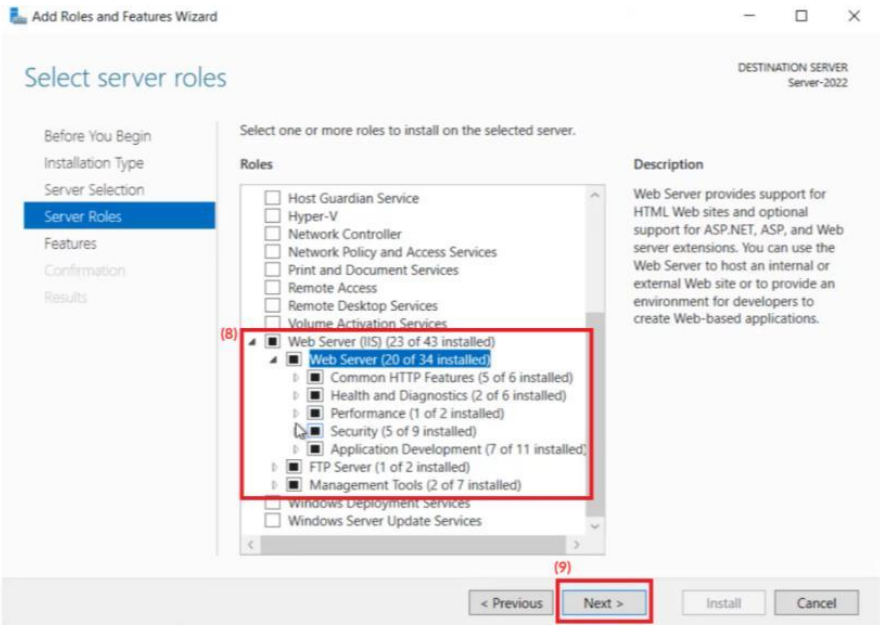


Figure 274: Detailed selection of Web Server (IIS) role

- (8) Go back to step 5 and check everything in “Web Server (IIS)”
- (9) Select “Next”

e. Install code

- Decompress 2 files sep490_fe.zip and sep490_api.zip.
- Open IIS Manager.

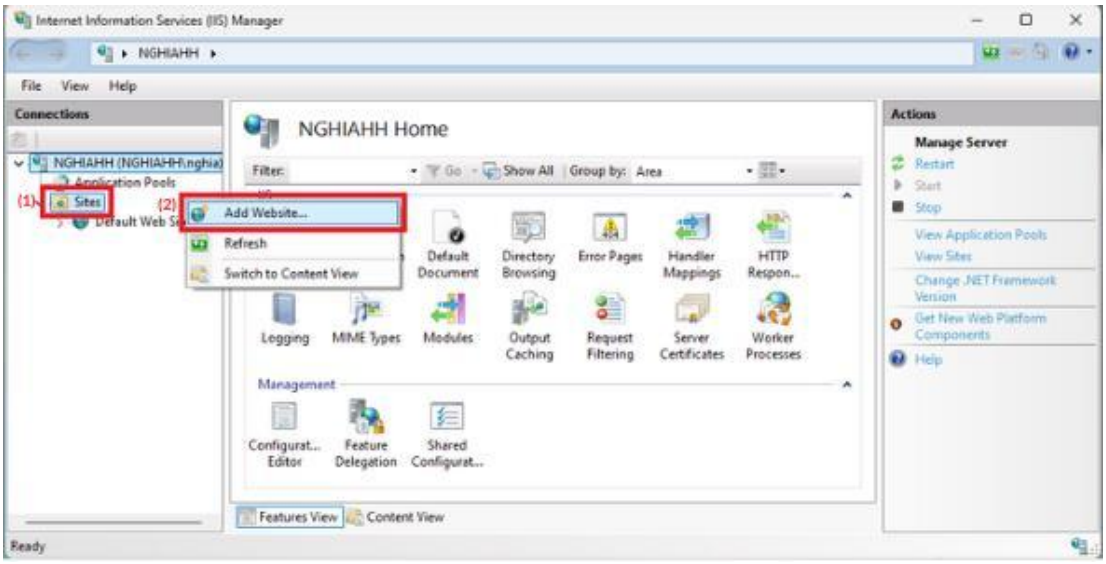


Figure 275: Adding a new website in Internet Information Services (IIS) Manager

- (1) Right-click and select “Sites”
- (2) Select “Add Website”

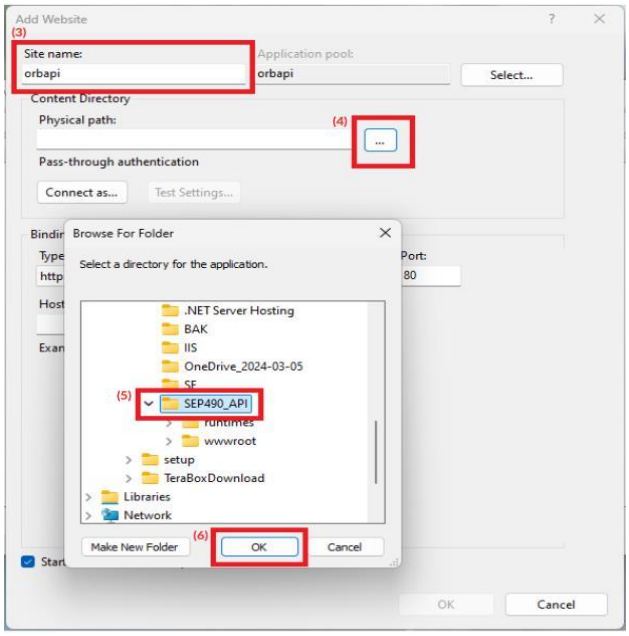


Figure 276: Selecting the "SEP490_API" folder

- (3) Enter “orbapi” in the Site name
- (4) Select “...”
- (5) Point to the extracted SEP490_API folder
- (6) Select “OK”

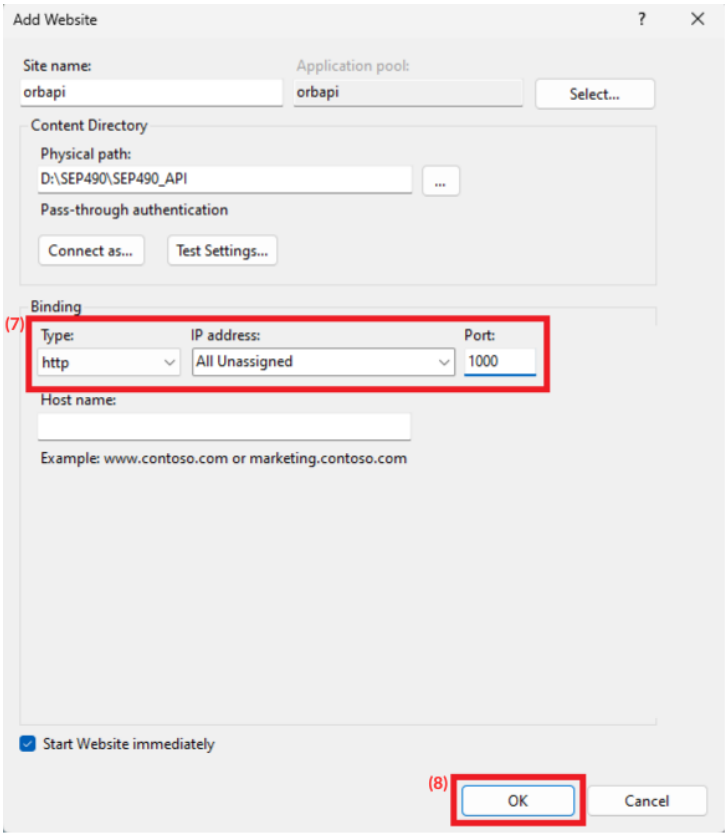


Figure 277: Configuring the "Add Website" dialog in IIS

- (7) Enter port 1000
- (8) Select “OK”

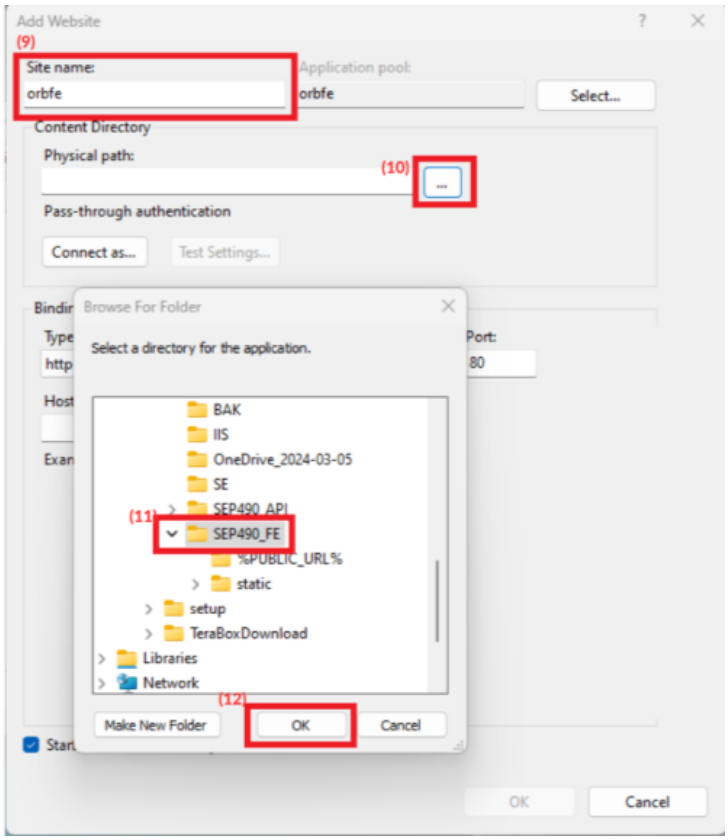


Figure 278: Selecting the "SEP490_FE" folder

- (9) Repeat steps 1 and 2 then enter “orbfe” into Site name
- (10) Select “...”
- (11) Point to the extracted SEP490_FE folder
- (12) Select “OK”

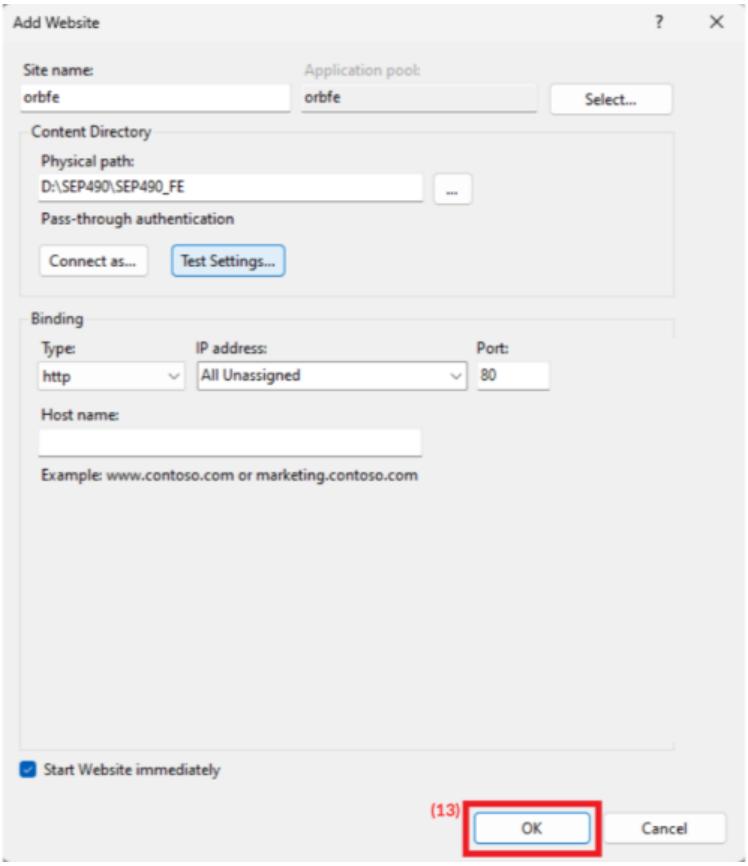


Figure 279: Configuring the "Add Website" dialog in IIS

- (13) “OK”

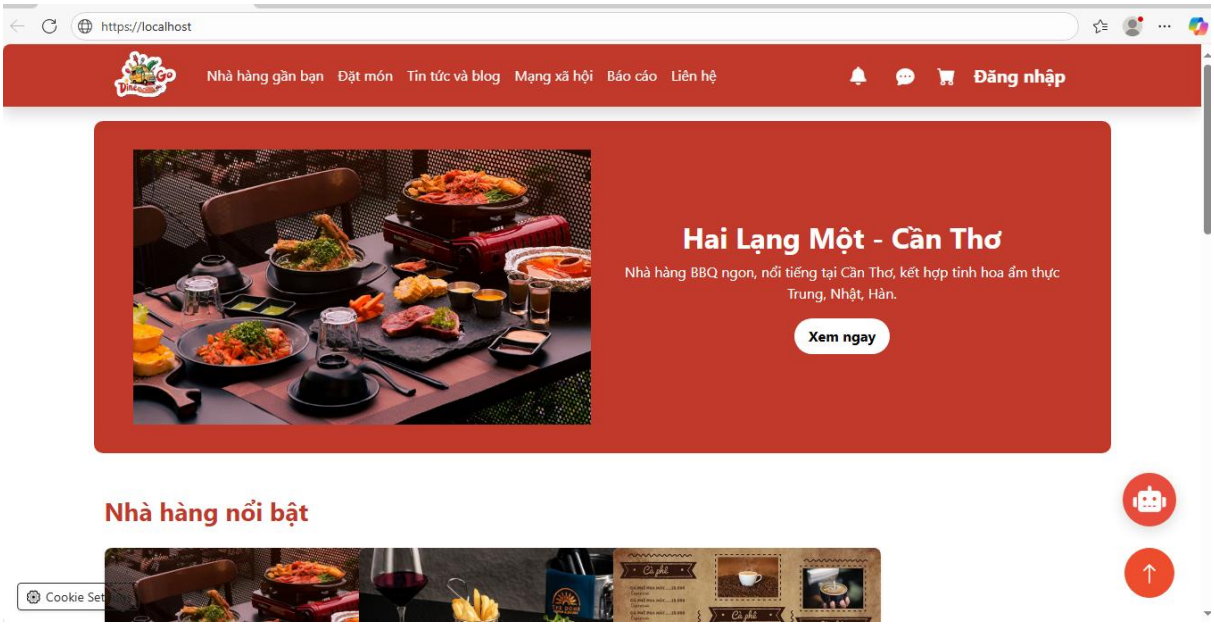


Figure 280: Web page

- Open browser and access to http://localhost/

3. User Manual

3.1 Terms and definitions

This table provides key terms and their definitions used in the DineGO system, clarifying the roles and responsibilities of different user types within the platform:

Table 26: Terms and definitions

No.	Term and definition	Description
1	Guest	A general visitor who browses the platform without creating an account. May search for restaurants, menus, or view ratings without interaction.
2	Customer	A registered user who can book tables, place orders, leave reviews, and manage personal account information.
3	Admin	The system administrator who manages user accounts, verifies restaurant owners, monitors activities, and ensures system stability.
4	Restaurant Owner	A business account holder who registers their restaurant, manages menu, confirms bookings, views customer feedback, and updates availability.

3.2 System requirements

Minimum hardware and software requirements for optimal system operation and compatibility:

- Hardware:
  - 2nd generation Core i5 (2 GHz+), 3rd/4th-generation Core i5 processor, or equivalent.
  - Windows 7 or later, Mac OS, Linux and Android devices. The system works best on Windows 10.
- Software:
  - Available web browsers: The system works best on Chrome. Alternative choices are Opera and Microsoft Edge.

3.3 Application Usage

a. Overview

There are 3 roles working in the system, divided into 2 main categories: Administrator and User (which includes both Customers and Restaurant Owners).

- Some of the functions are shared among all users:
  1. Login/Logout
  2. View/Update profile
  3. Change password
  4. View notifications
  5. View blog
  6. Forget password
- As an administrator, the user can have the right to access the following features:
  1. Manage customer accounts (Create, Edit, Delete, Block/Unblock, Activate)
  2. Manage restaurant owner (Create, Edit, Delete)
  3. Manage business license verification (Review, Approve, Reject)
  4. Manage restaurants

5. Manage restaurant categories
6. Manage food orders (View, Process all orders)
7. View table reservation
8. Manage payments (View payments, print invoice)
9. Manage blog
10. Manage vouchers
11. Manage notifications
12. View dashboard reports and statistics
13. Manage system logs (View activity logs)

- As a customer, the user can have the right to access the following features:
    1. Search restaurants
    2. View restaurant detail (Menu, ratings, information)
    3. Browse food menus (View food items, prices)
    4. Manage shopping cart (Add, Remove, Update quantities)
    5. Track order delivery
    6. Make table reservations
    7. View reservation history
    8. Create social posts
    9. Comment on posts
    10. Like posts
    11. Chat with friends
    12. Manage friends (Add, Accept, Remove friends)
    13. Register as restaurant owner (Apply to become owner)
    14. Manage voucher
- As a restaurant owner, the user can have the right to access the following features:
    1. Create restaurant profiles
    2. Update restaurant information
    3. Manage menu
    4. Manage food items
    5. Handle table reservations
    6. Process food orders
    7. Respond to customer inquiries
    8. View statistics dashboard
    9. View business license
    10. Submit business license

b. Register

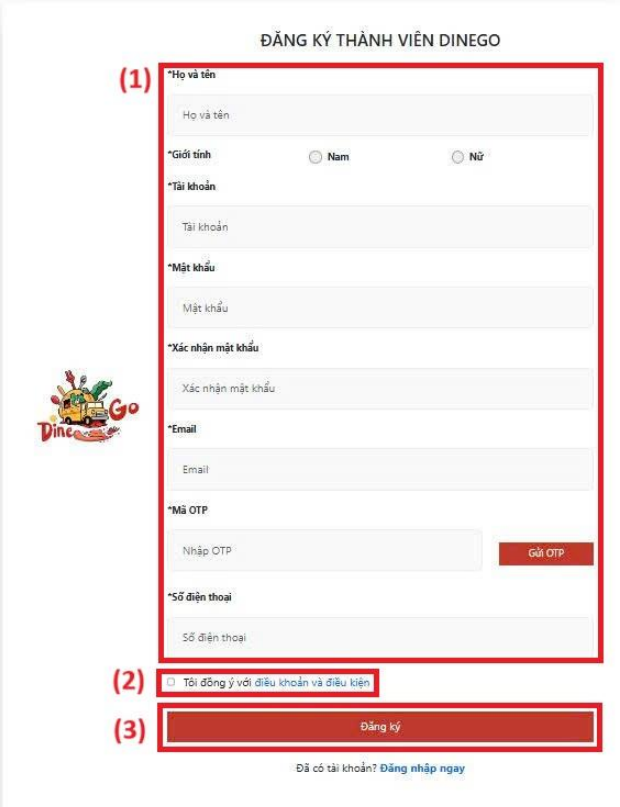


Figure 281: Register page of DineGO

This page allows users to create a new account

- (1) Fill in personal information in the required fields
- (2) Tick the checkbox “Tôi đồng ý với điều khoản và điều kiện” to proceed
- (3) Click the [Đăng ký] button to register
    - If any required field is empty, the system will show an error message.
    - If the “Email” or “Số điện thoại” is in an invalid format, a warning will be shown.
    - If the “Mật khẩu” and “Xác nhận mật khẩu” do not match, an error will appear.
    - If the user does not agree to the “điều khoản và điều kiện”, the registration will be blocked.
    - If the “Tài khoản” or “Email” already exists, the system will return an appropriate error message.

c. Login

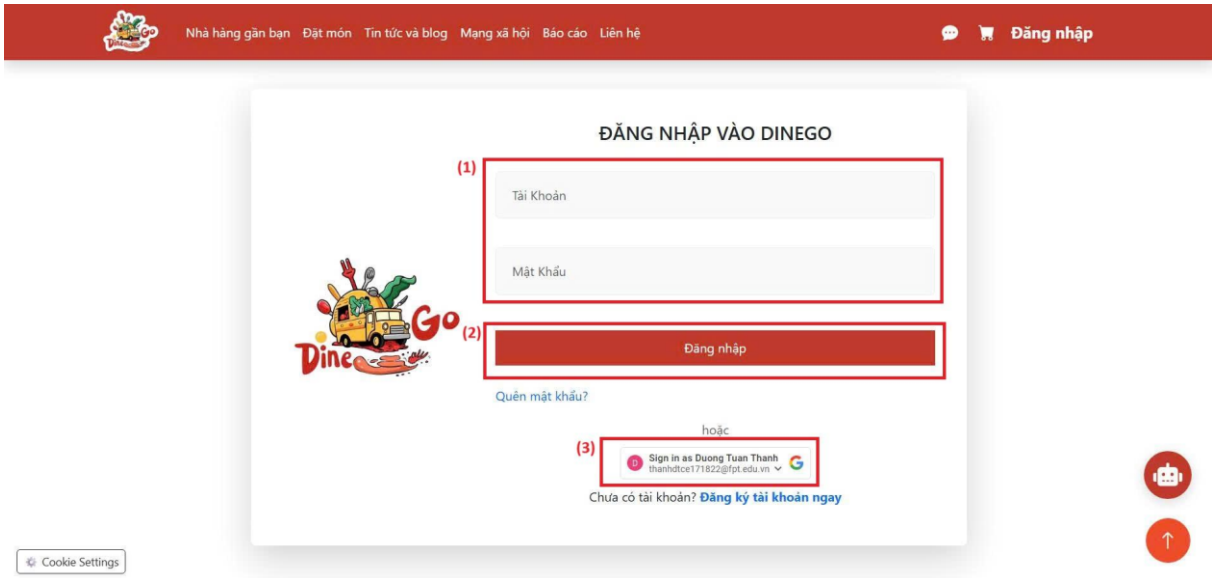


Figure 282: Login page of DineGO

This page allows users to log in account using either a traditional login form or via Google account

- (1) Enter login credentials
- (2) Click the [Đăng nhập] button
- (3) Or choose Google Sign-in

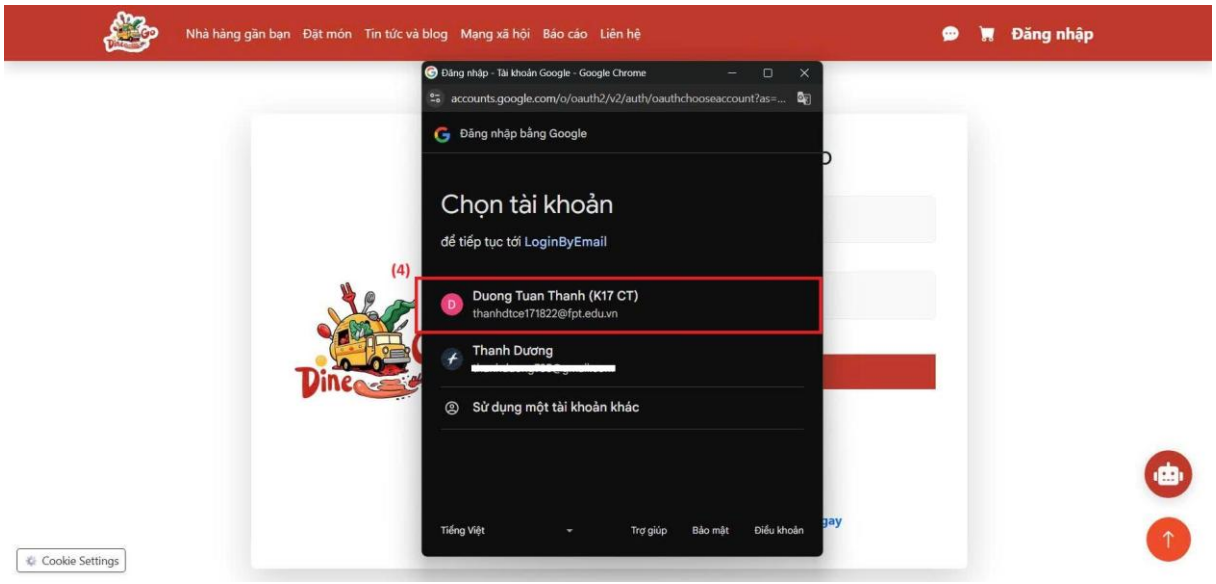


Figure 283: Login with Google

- (4) Select your Google account
  - If the “Tài khoản” or “Mật khẩu” field is left empty, an error message will be shown.

- If the tài khoản/mật khẩu format is invalid, an error will appear.
- If the credentials are incorrect, the system will notify that the account does not exist.
- If using Google login, you must choose a valid account to proceed. If the Google account is not registered in the system, the login may be rejected or redirected to a registration process.

d. Admin login

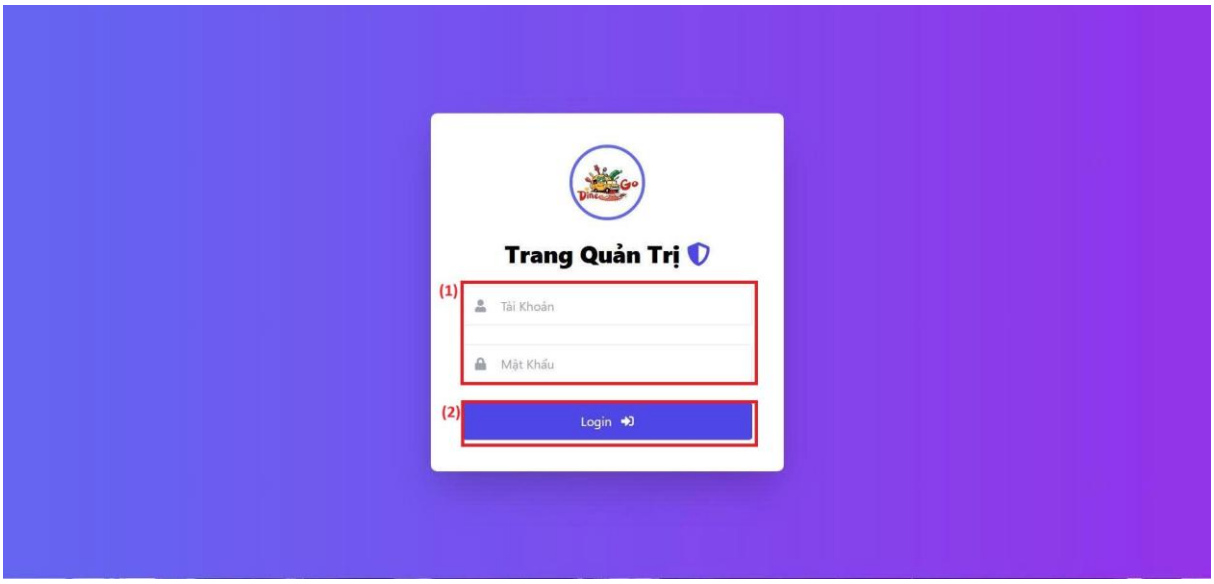


Figure 284: Login admin page of DineGO

This page allows administrators to log in

- (1) Enter login credentials
- (2) Click the [Đăng nhập] button
  - If either the “Tài khoản” or “Mật khẩu” field is left blank, the system will show an error message.
  - If the credentials are incorrect, the system will display an authentication failure message.
  - Only users with admin rights can log in through this page; regular user accounts will be denied access

e. View/update profile, change avatar

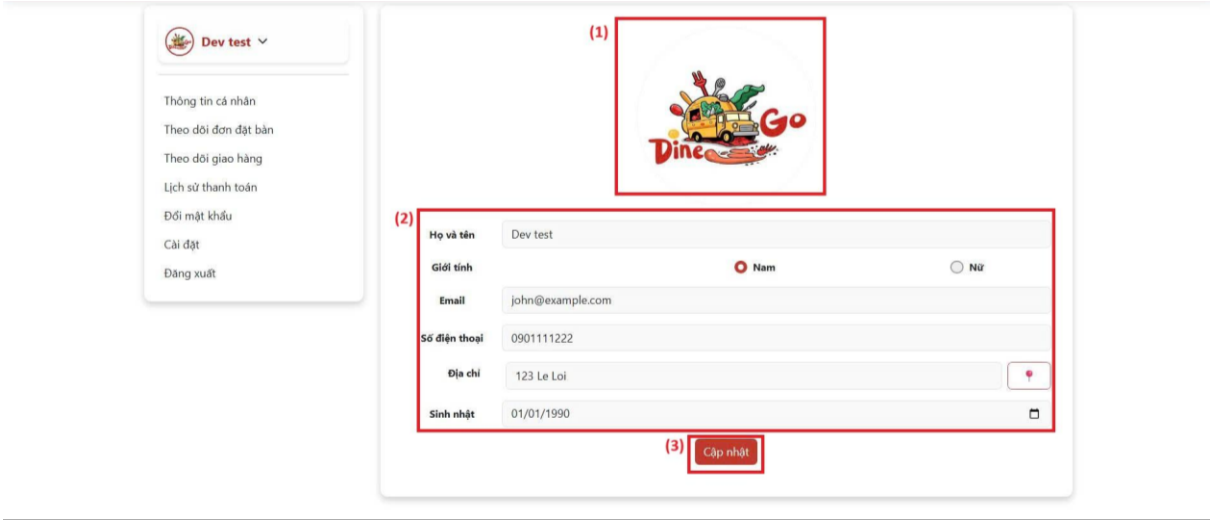


Figure 285: Profile page of DineGO

This page allows users to view and update their personal information and profile picture

- (1) View and update profile image
- (2) Edit personal information
- (3) Click the [Cập nhật] button
    - Uploaded image must be in a valid format (e.g., .jpg, .png) and within size limits.
    - "Email" and "Số điện thoại" must follow correct formats.
    - Empty or invalid fields will trigger error messages.
    - After a successful update, a success message or toast notification will be displayed.

f. Change password

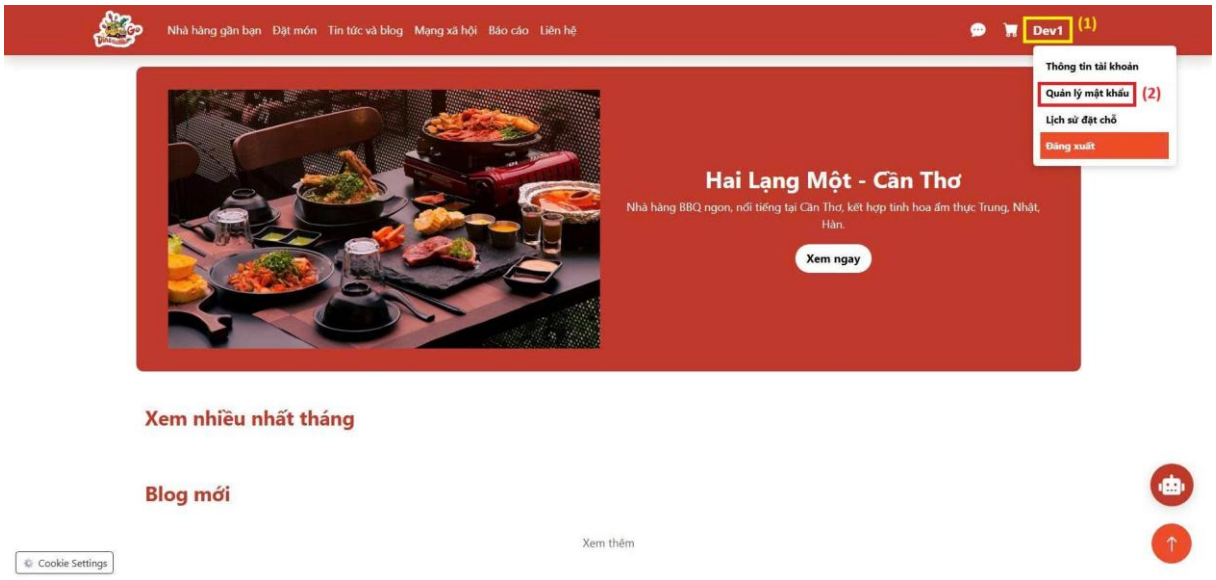


Figure 286: Home page of DineGO

This feature allows users to securely update their account password within the personal profile section

- (1) Hovering the account section shows the action on of the account.
- (2) Click [Quản lí mật khẩu] button on for redirecting to the change password page.

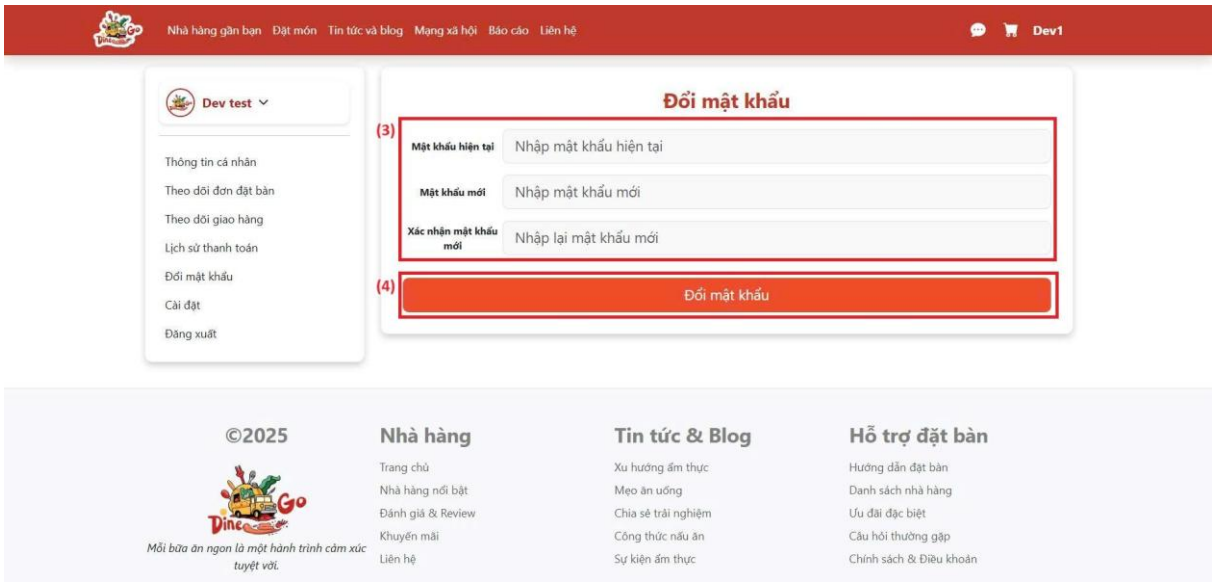


Figure 287: Change password page

- (3) Fill in the password change form
- (4) Click the [Đổi mật khẩu] button
    - All fields are required.
    - The “Mật khẩu hiện tại” must match the one stored in the system.
    - The “Mật khẩu mới” and “Xác nhận mật khẩu mới” must be identical.
    - If validation fails an appropriate error message will be displayed.
    - Upon successful update, users are notified with a success alert.

g. Logout

i. Customer logout

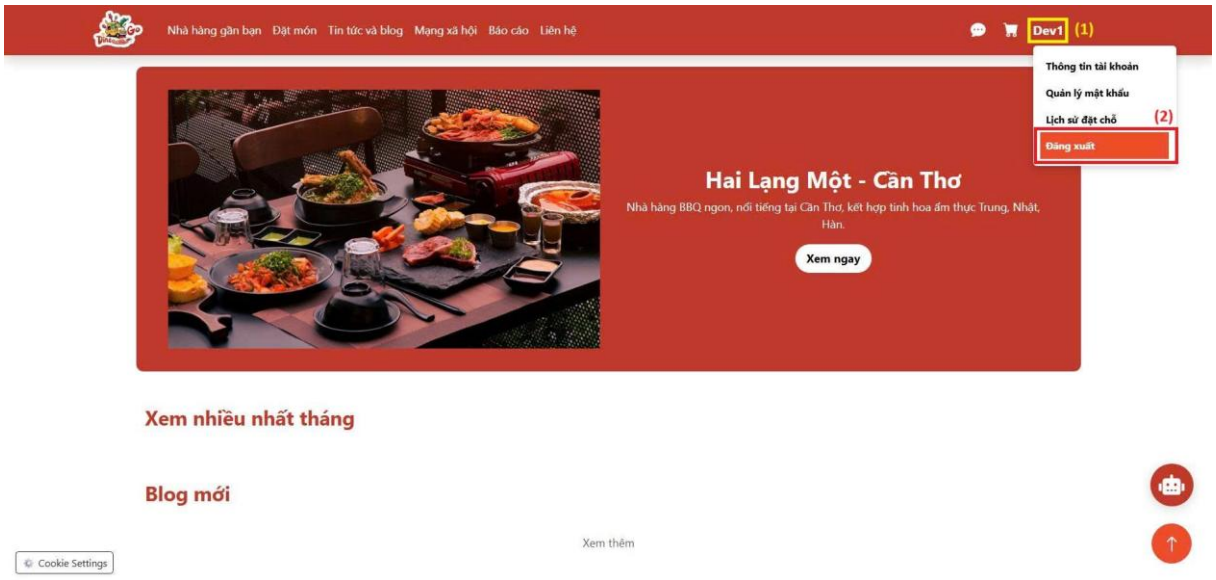


Figure 288: Logout function of DineGO

This function allows users to safely log out

- (1) Hovering the account section shows the action on of the account.

- (2) Click [Đăng xuất] button on for logout.

ii. Admin logout

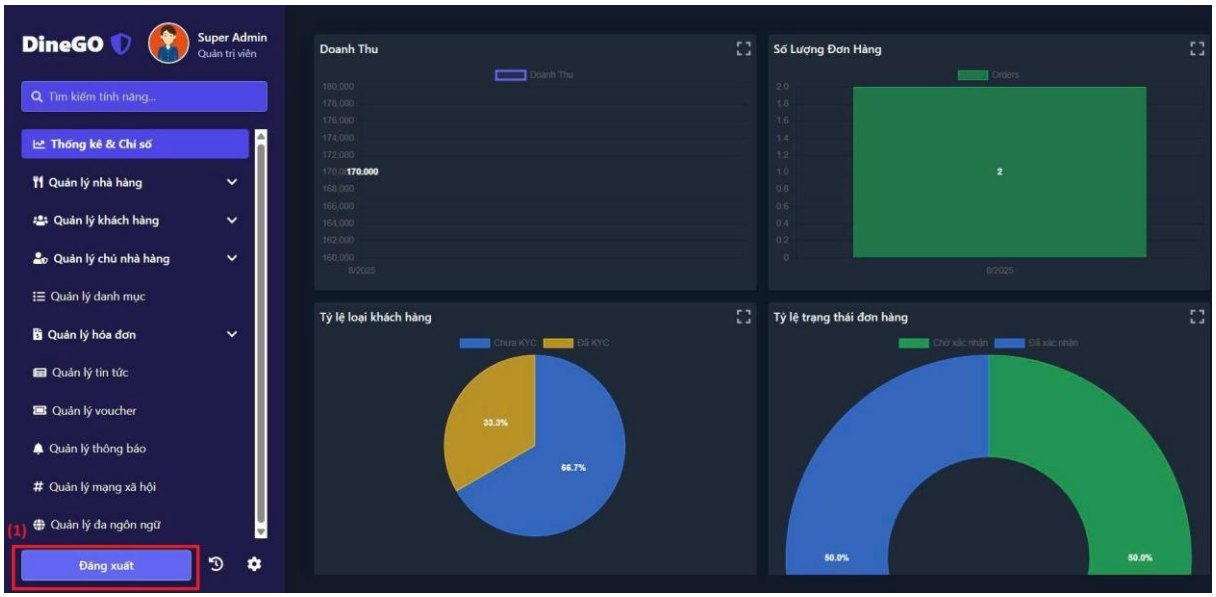


Figure 289: Admin logout function of DineGO

This function allows admin users (e.g., Super Admin) to securely sign out

- (1) Click the [Đăng xuất] button in the sidebar

h. View blog

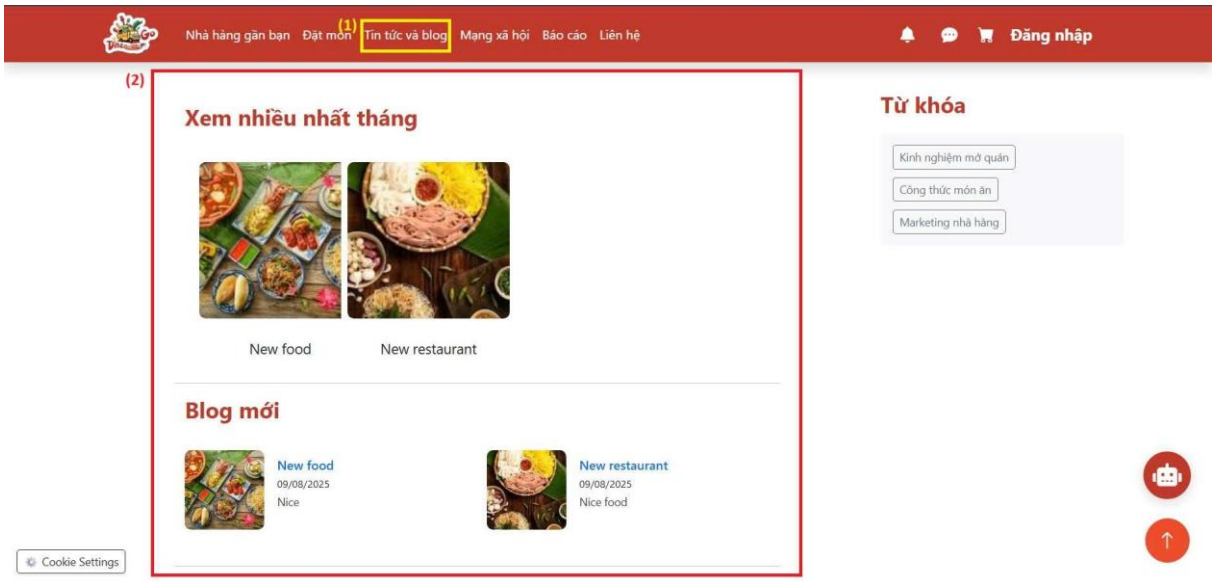


Figure 290: Forgot password function

This feature allows users to browse blog posts about restaurants and food.

- (1) Navigate to [Tin tức và blog]
- (2) Browse Blog Content
    - The page should load all relevant blog posts from the database.
    - Images must be displayed correctly for each post.
    - The most viewed section should be updated automatically based on actual view counts.
    - Blog posts must be sorted by publish date in the “Blog mới” section.
    - Clicking on a blog title or image must navigate to the blog detail page.

i. Forgot password

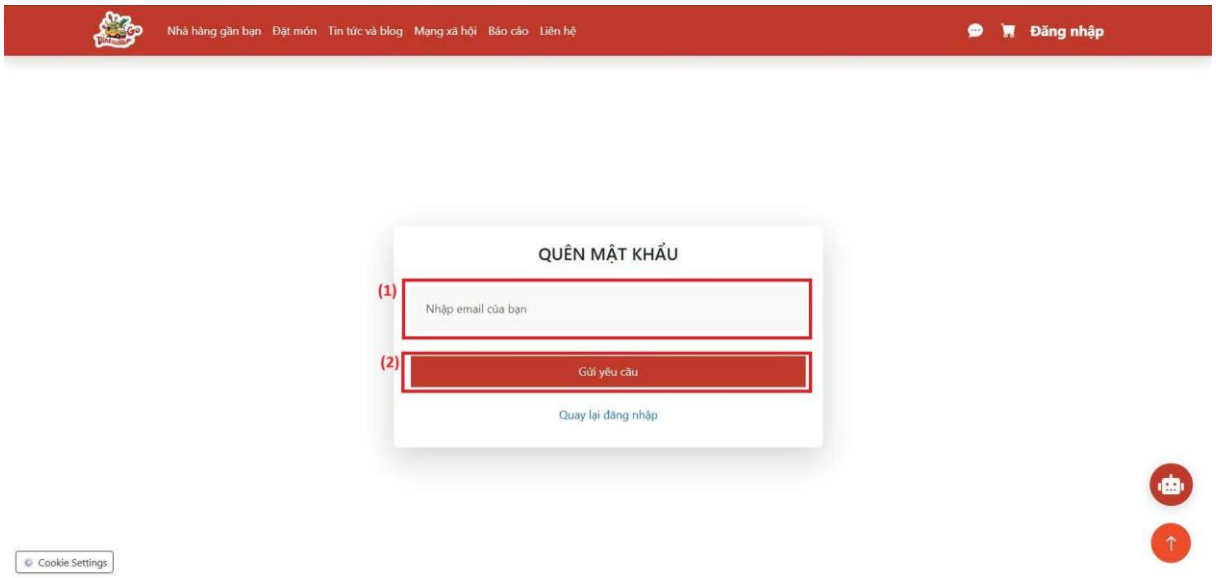


Figure 291: Forgot password function

This feature allows users to reset their account password when they forget it

- (1) Enter your email address
- (2) [Gửi yêu cầu] button to submit

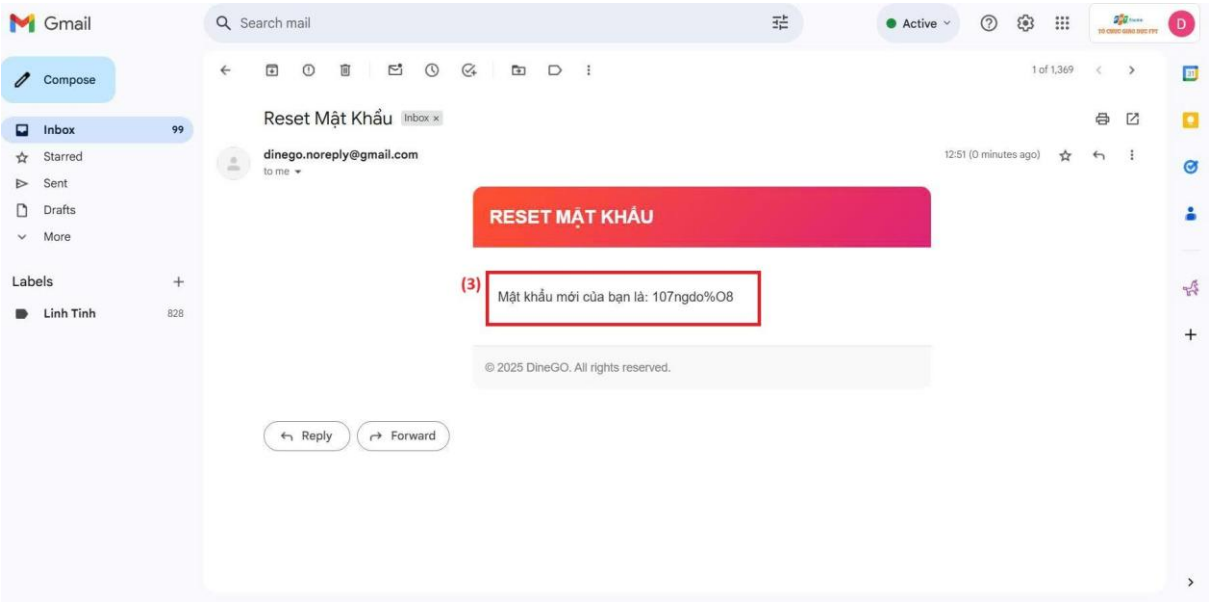


Figure 292: Gmail reset password

- (3) Check your email for the new password
  - The new password is automatically generated by the system.
  - Users should change the generated password immediately after logging in.
  - If the user does not receive the email, they should check the spam folder or request again.

3.3.1 Admin feature

a. Manage customer account

i. View customer

Figure 293: View customer list of DineGO

This feature allows the Admin to view all customers registered

- (1) [Danh sách khách hàng] menu item to open the customer list page.
- (2) Customer list table displaying

- All displayed data must be retrieved accurately from the database.
- Status must reflect the latest account verification state.
- If the list loads successfully, all customers are shown in the table.

ii. Create customer

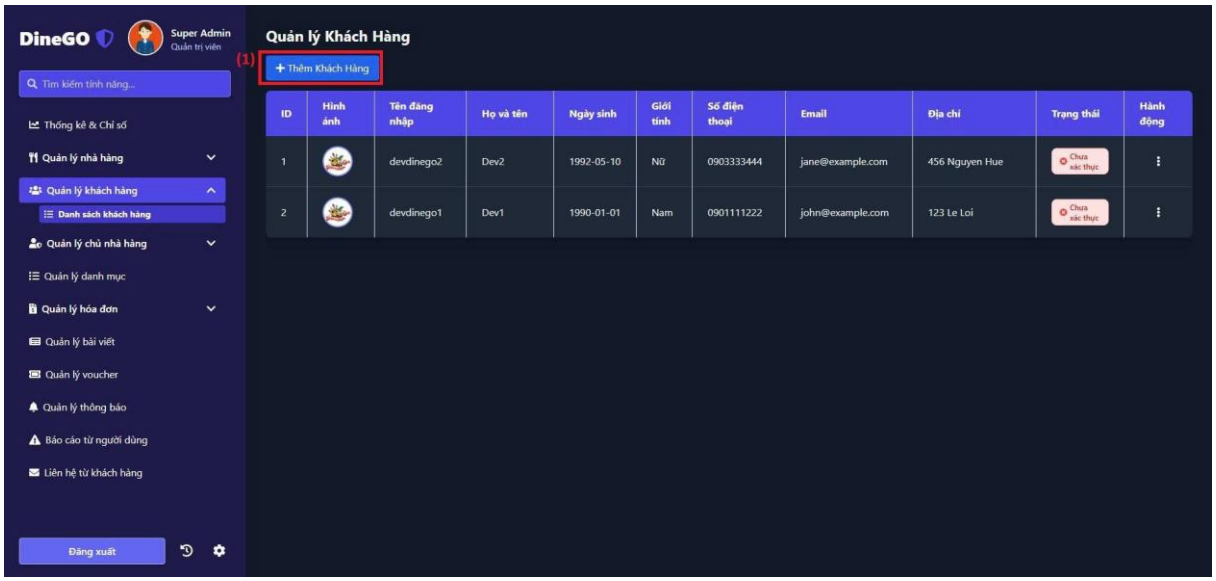


Figure 294: Customer list of DineGO

This feature allows the Admin to add a new customer to the system.

- (1) [Thêm Khách Hàng] button to open the add customer form.

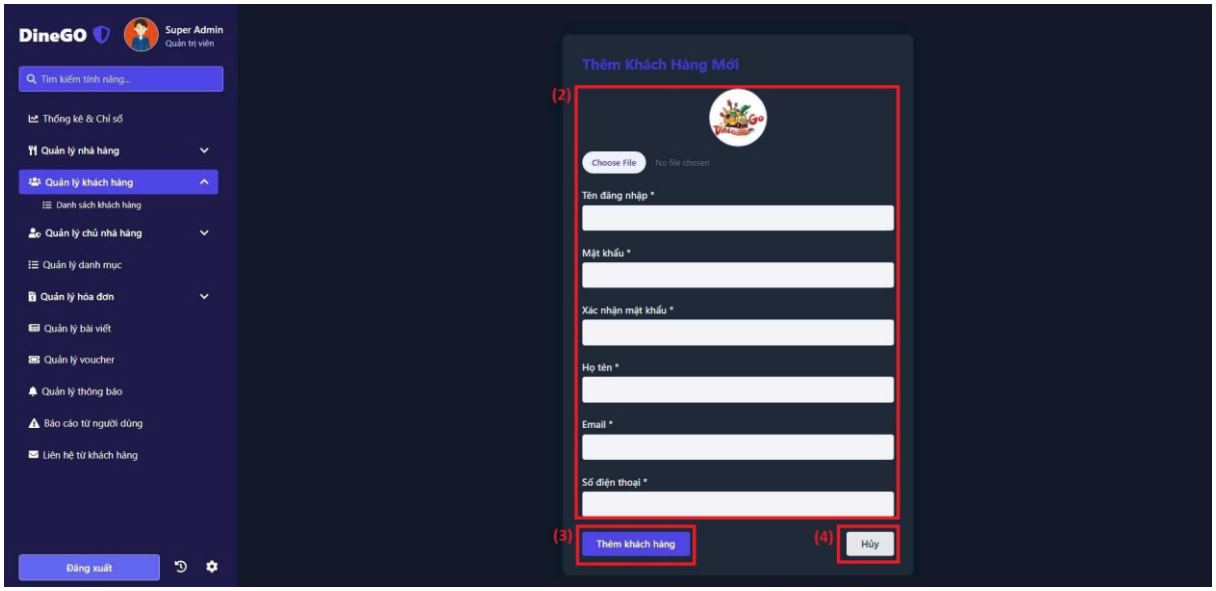


Figure 295: Add customer form of DineGO

- (2) Fill all fields
- (3) [Thêm khách hàng] button to save the new customer.
- (4) [Hủy] button to return to the customer list without saving.
  - "Tên đăng nhập" is required and must be unique.
  - "Mật khẩu" is required and must meet the minimum length requirement.

- “Xác nhận mật khẩu” must match the “Mật khẩu” field.
- “Email” must be in a valid email format.
- “Số điện thoại” must be 10–11 digits, numbers only.
- Uploaded image must be within allowed size and format (JPG/PNG).
- On success, the new customer appears in the list.
- On failure, an error message is displayed.

iii. Update customer

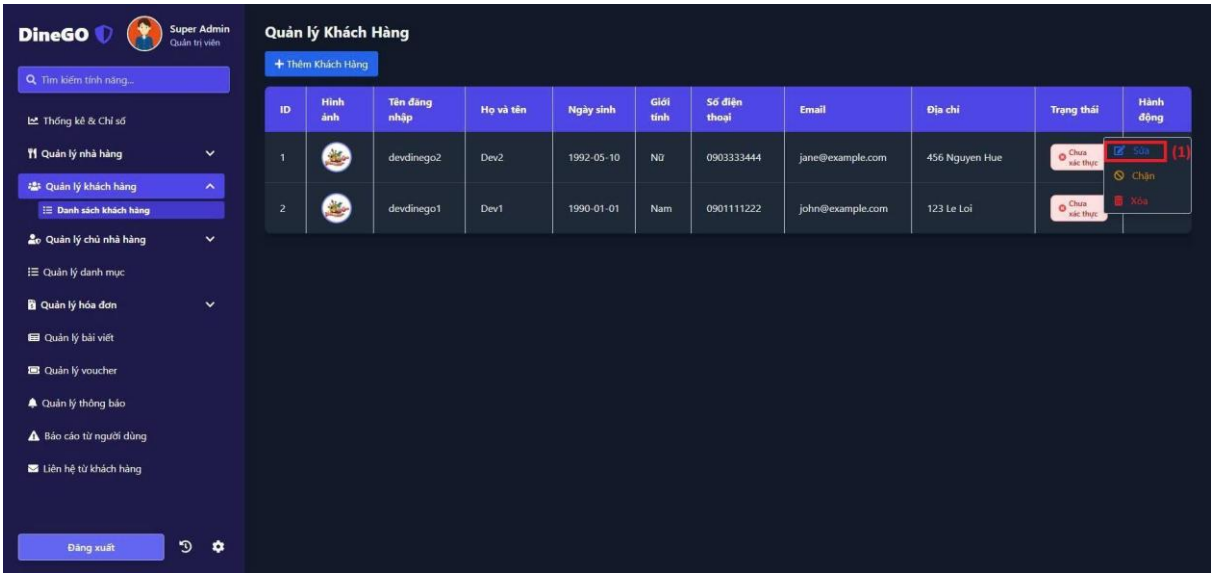


Figure 296: Customer list of DineGO

This feature allows the Admin to update an existing customer’s information.

- (1) [Sửa] icon in the Hành động column of the customer list to open the edit customer form

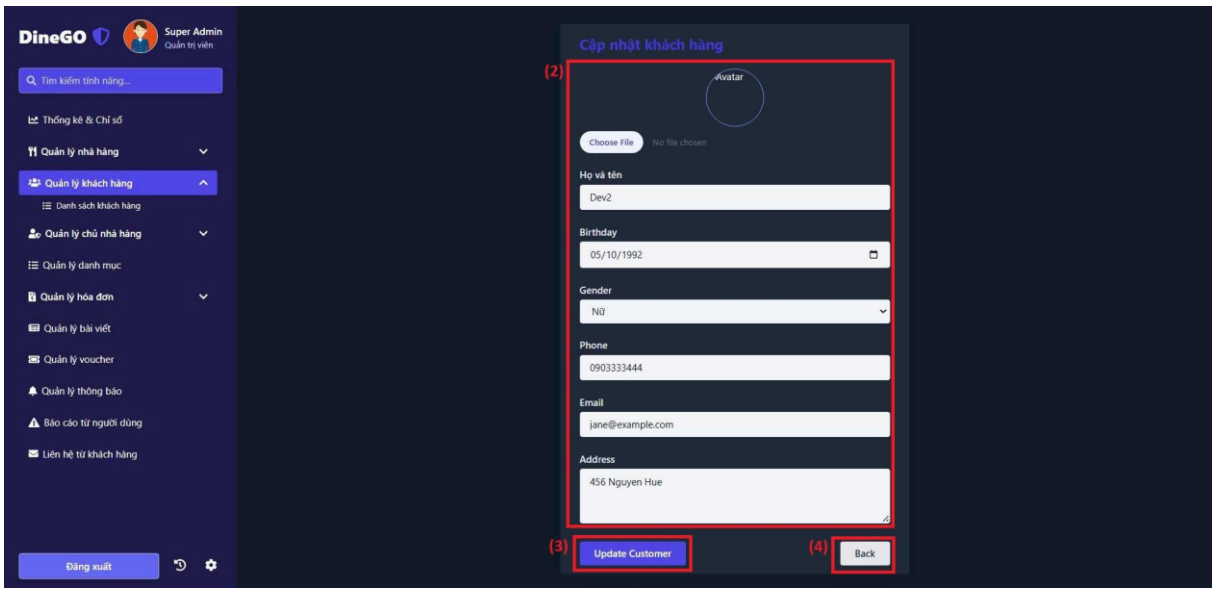


Figure 297: Edit customer form of DineGO

- (2) Enter the updated customer information to want update
- (3) [Update Customer] button to save changes.

- (4) [Back] button to return to the customer list without saving.
    - “Họ và tên” is required.
    - “Số điện thoại” must be valid (10–11 digits, numbers only).
    - “Email” must be in a valid email format.
    - Uploaded image must be within the allowed size and format
    - If the update is successful, the customer’s detail are updated in the list.
    - If the update fails, an error message is displayed.

iv. Block customer

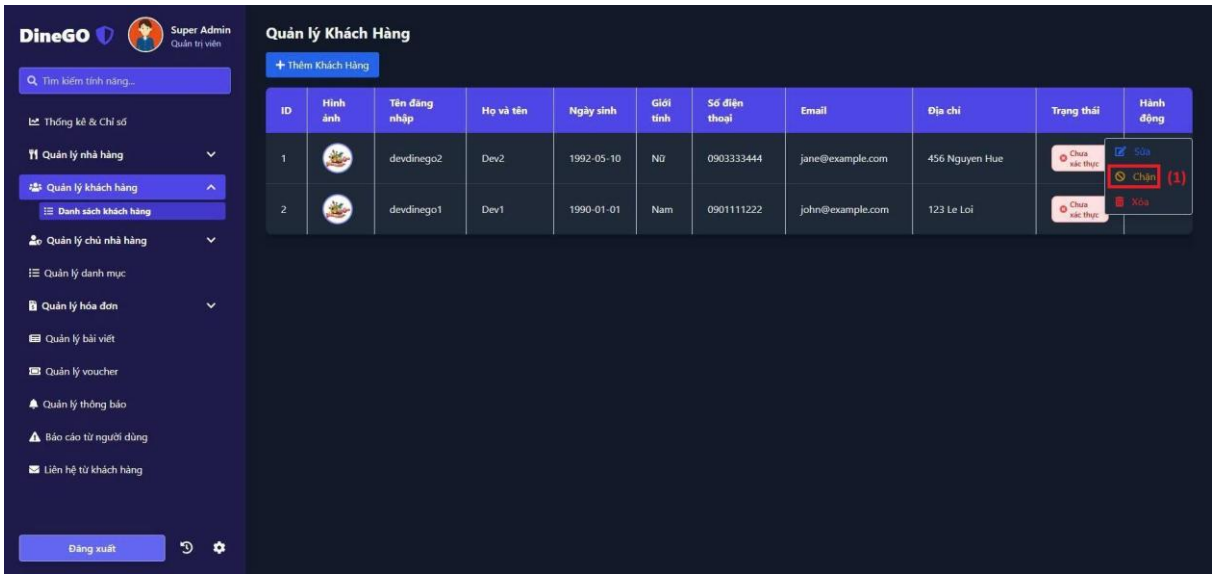


Figure 298: View customer list of DineGO

This feature allows the Admin to block a customer

- (1) [Chặn] icon in the Hành động column of the customer list to open the block customer form.

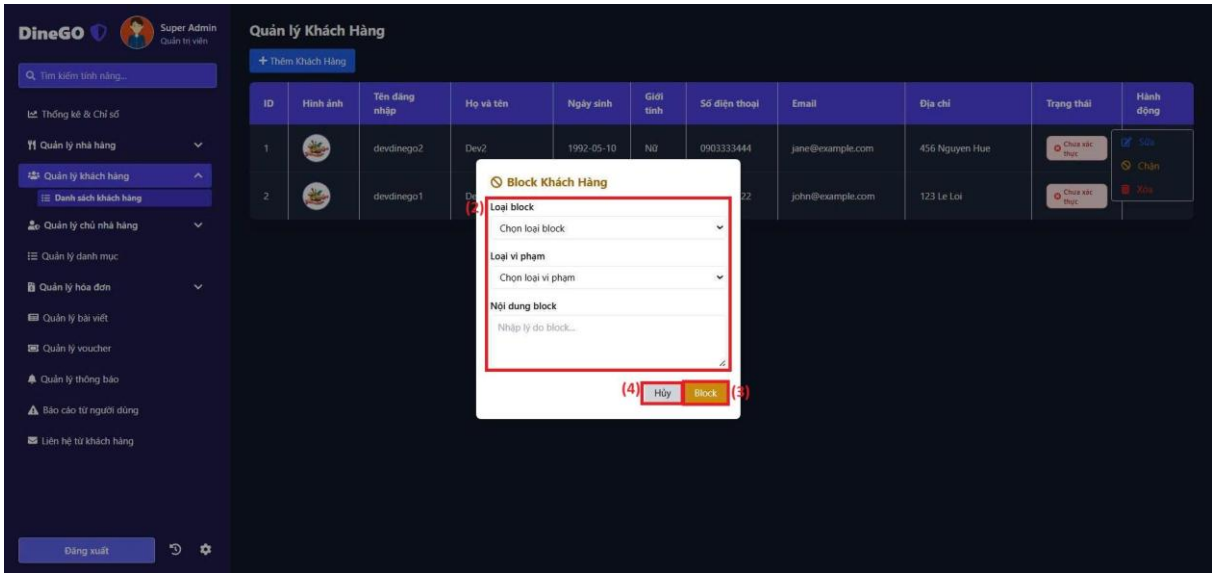


Figure 299: Block form of DineGO

- (2)Fill all fields
- (3) [Block] button to confirm and apply the block.
- (4) [Hủy] button to cancel and close the form without blocking.
    - “Loại block” is required.
    - “Loại vi phạm” is required.
    - “Nội dung block” is required and must be at least 5 characters.
    - If blocking is successful, the customer’s status changes to Blocked in the list.
    - If blocking fails, an error message is displayed.

v. Unblock customer

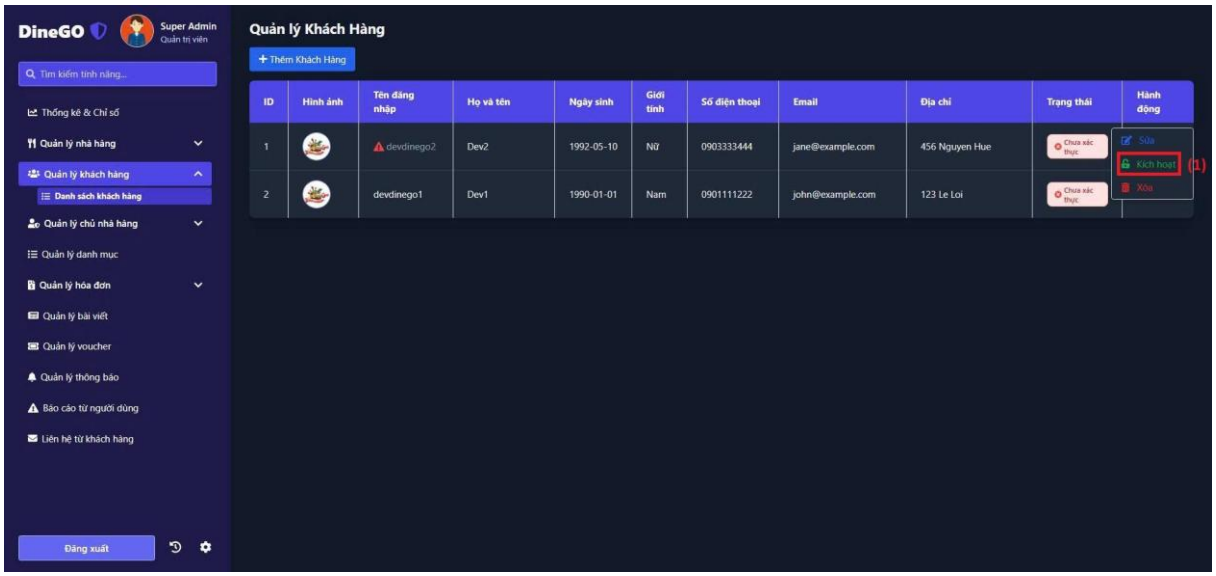


Figure 300: View customer list of DineGO

This feature allows the Admin to activate a previously blocked customer account.

- (1) [Kích hoạt] icon in the Hành động column of the customer list to open the activation form.

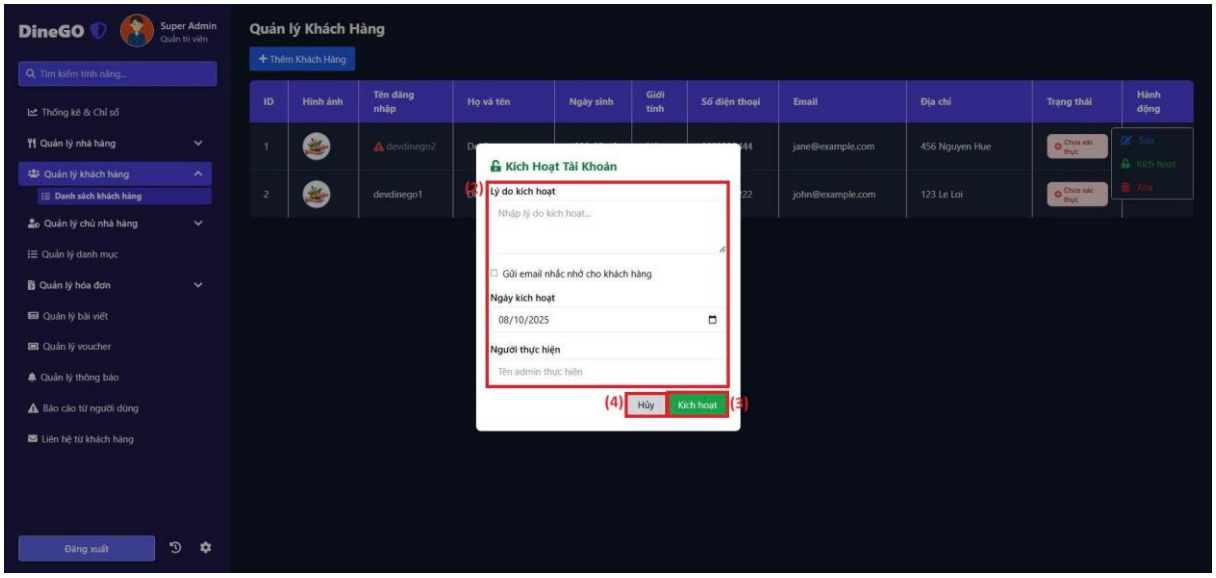


Figure 301: Unblock form of DineGO

- (2) Fill all fields
- (3) [Kích hoạt] button to confirm and activate the account.
- (4) [Hủy] button to cancel and close the form without activating.
    - Ngày kích hoạt cannot be in the past.
    - If activation is successful, the customer's status changes to Active in the list.
    - If activation fails, an error message is displayed.

b. Manage restaurant owner

i. View restaurant owner

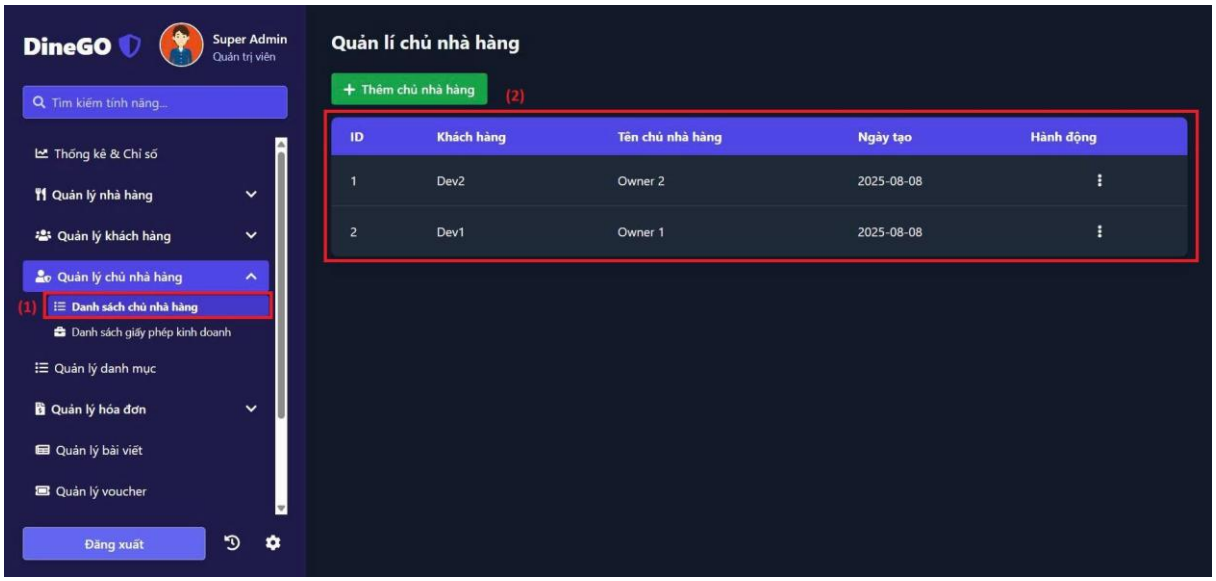


Figure 302: View restaurant owner list of DineGO

This feature allows the Admin to view and manage all restaurant owners

- (1) [Danh sách chủ nhà hàng] menu item to open the list page.
- (2) Table listing restaurant owners
    - Admins can review owner detail

ii. Create restaurant owner

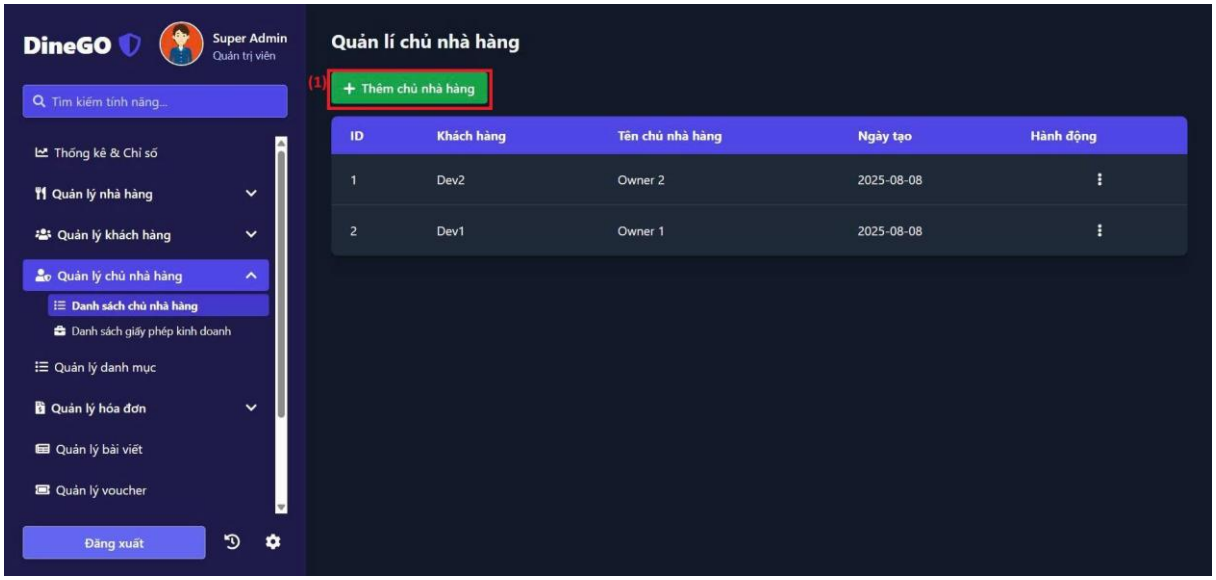


Figure 303: View restaurant owner page of DineGO

This feature allows the Admin to create a new restaurant owner.

- (1) [Thêm chủ nhà hàng] button to open the creation form.

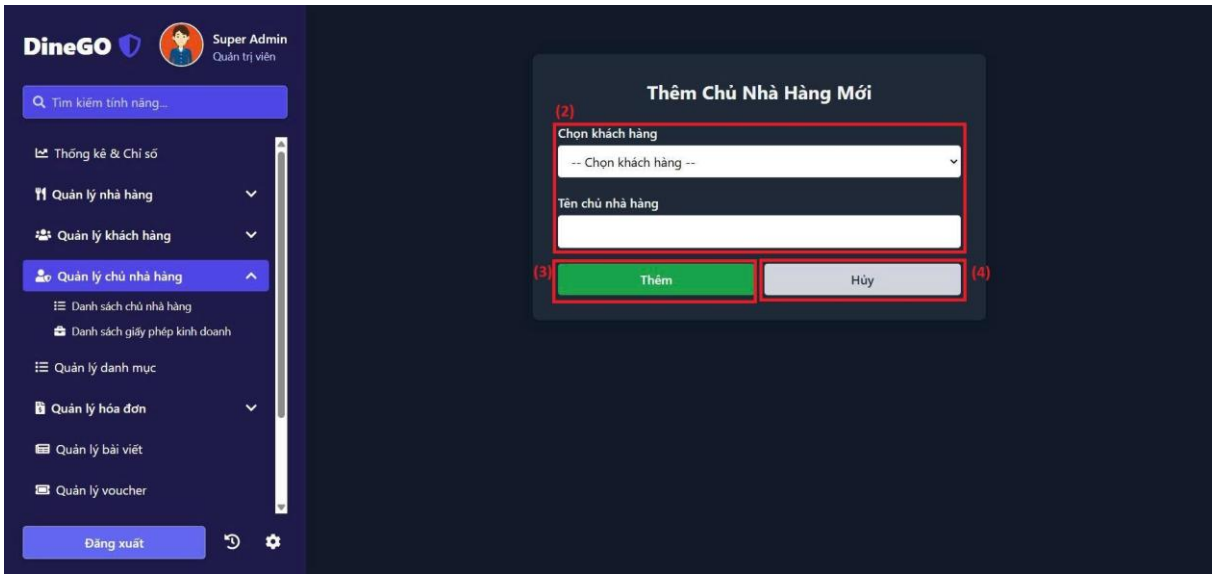


Figure 304:. Add restaurant owner page of DineGO

- (2) Choose an existing customer and enter the restaurant owner's name.
- (3) [Thêm] button to confirm and save the new restaurant owner.
- (4) [Hủy] button to abort the creation process.
    - "Chọn khách hàng" is required.
    - "Tên chủ nhà hàng" must not be empty.
    - If creation is successful, the new owner appears in the list.
    - If creation fails, an error message should be displayed.

iii. Update restaurant owner

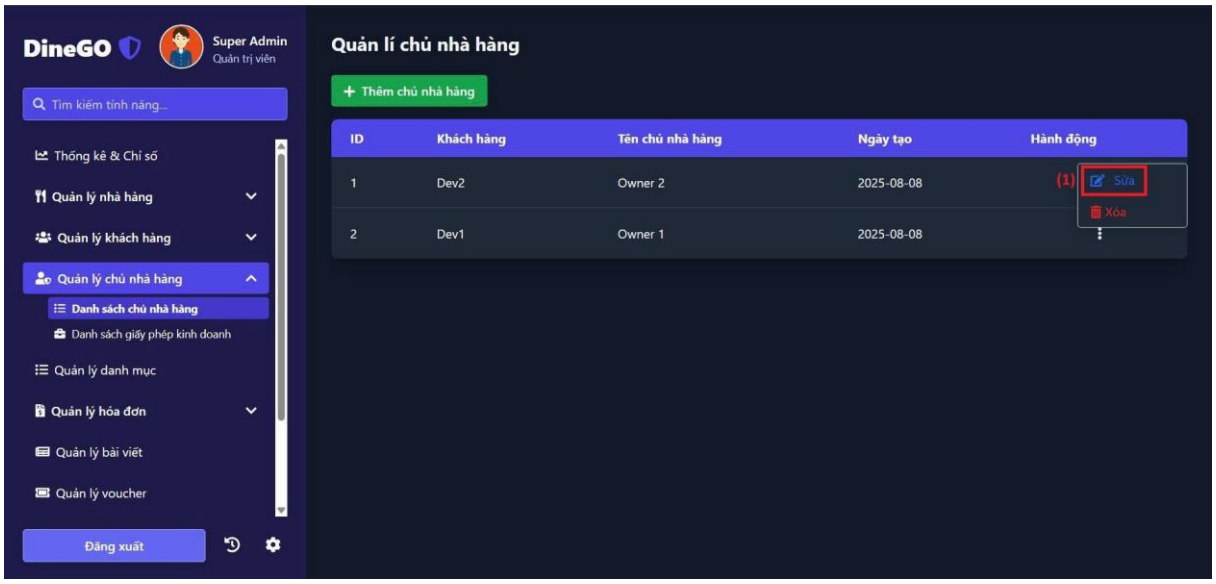


Figure 305: View restaurant owner page of DineGO

This feature allows the Admin to edit an existing restaurant owner's information.

- (1) [Edit icon] next to the restaurant owner you want to update.

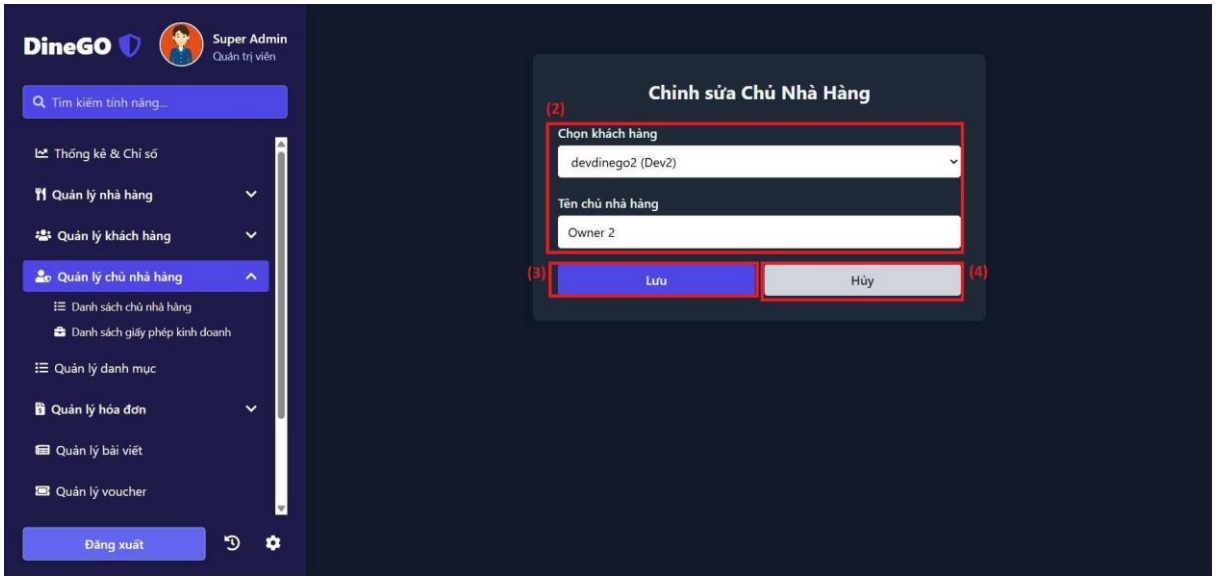


Figure 306: Update restaurant owner page of DineGO

- (2) [Chọn khách hàng] dropdown to select or change the associated customer and "Tên chủ nhà hàng"to update the name.
- (3) [Lưu] button to confirm and save changes.
- (4) [Hủy] button to abort the update process.
    - "Tên chủ nhà hang" must not be empty.
    - "Chọn khách hàng" must be valid and active in the system

iv. Delete restaurant owner

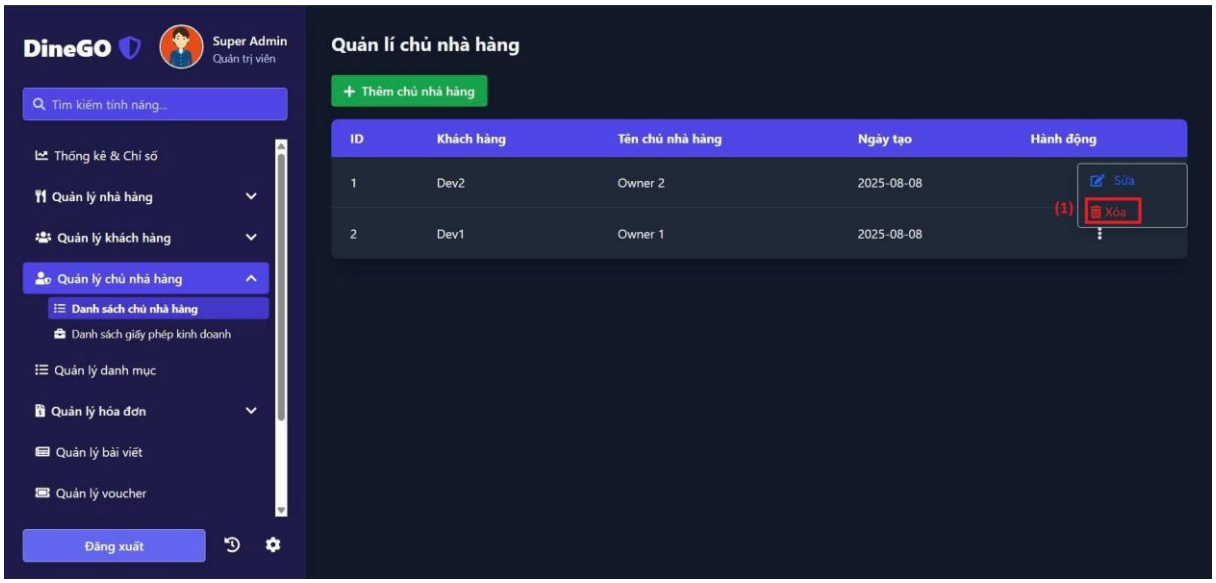


Figure 307: View restaurant owner page of DineGO

This feature allows the Admin to remove an existing restaurant owner from the system.

- (1) [Delete icon] next to the restaurant owner you want to remove.

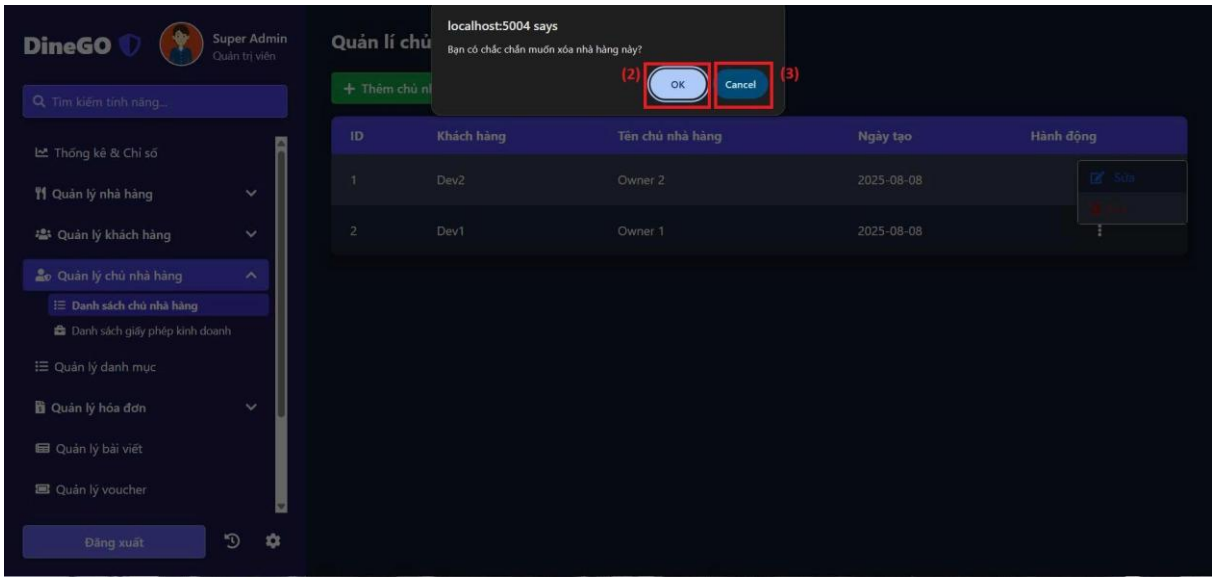


Figure 308: View restaurant owner page of DineGO

- (2) [OK] button to confirm deletion.
- (3) [Cancel] button to abort the deletion process.
    - If the restaurant owner is linked to active restaurants or ongoing reservations, deletion should be blocked, and an error message should be displayed.
    - If deletion is successful, the restaurant owner will be removed from the list.
    - If deletion fails due to a system error, an appropriate error message should be displayed.

c. Manage business license verification

i. Review business license

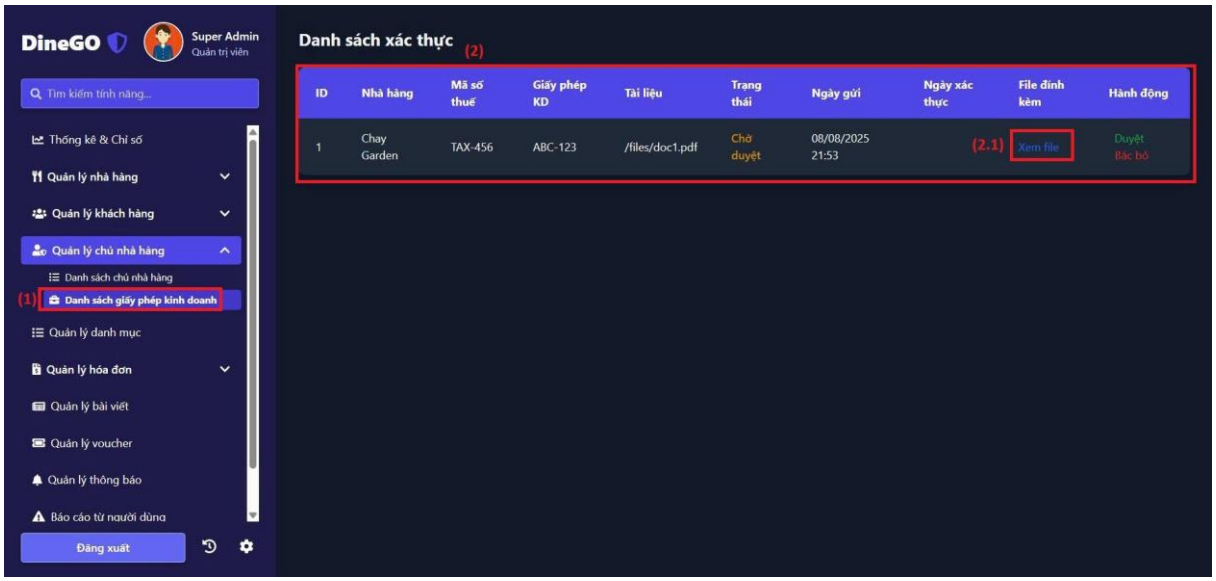


Figure 309: Business license page of DineGO

This feature allows the Super Admin to review and verify business license submissions from restaurant owners.

- (1) [Danh sách giấy phép kinh doanh] menu option in the admin panel to open the verification list
- (2) Verification Table – Displays all pending, approved, or rejected business license requests with the following columns:
- (2.1) [Xem file] – Opens the uploaded business license document for review.
    - The tax code and business license number must not be empty.
    - A request can only be approved or rejected once; after that, the action buttons are disabled.
    - Clicking [Duyệt] button changes the status to "Đã duyệt" and records the verification date.
    - Clicking [Bác bỏ] button changes the status to "Bị từ chối" and records the verification date.

ii. Approve business license

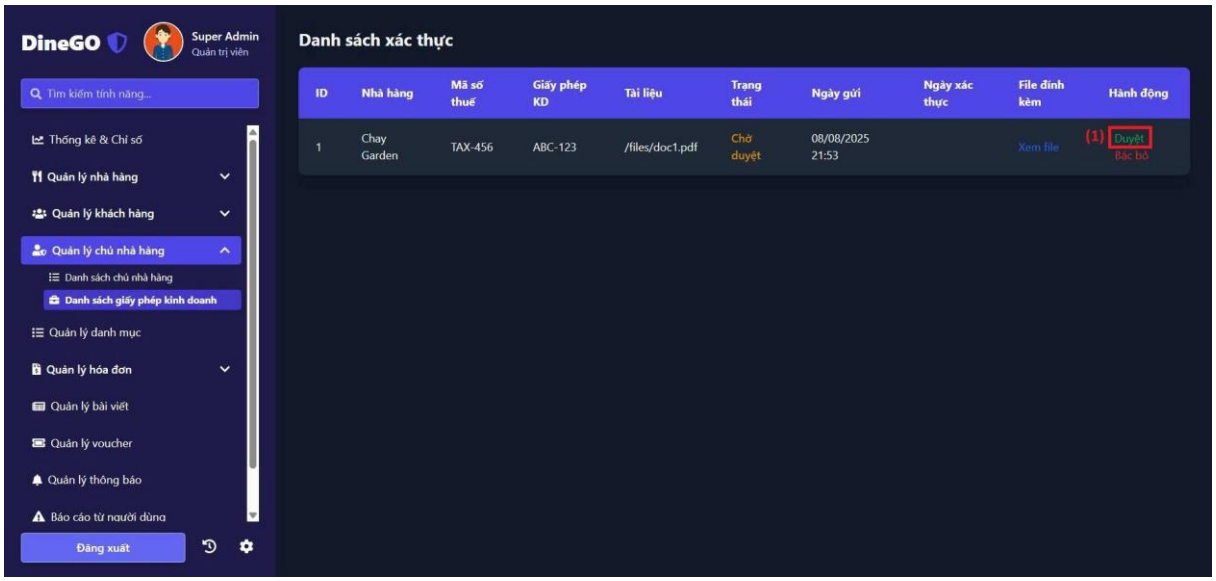


Figure 310: Business license page of DineGO

This feature allows the Admin to approve a pending business license submission from a restaurant owner.

- (1) [Duyệt] approves the selected business license request.
    - The request must be in "Chờ duyệt" status before it can be approved.
    - Once approved, the status changes to "Đã duyệt" and cannot be changed back.
    - The system must record the date and time of approval.

iii. Reject business license

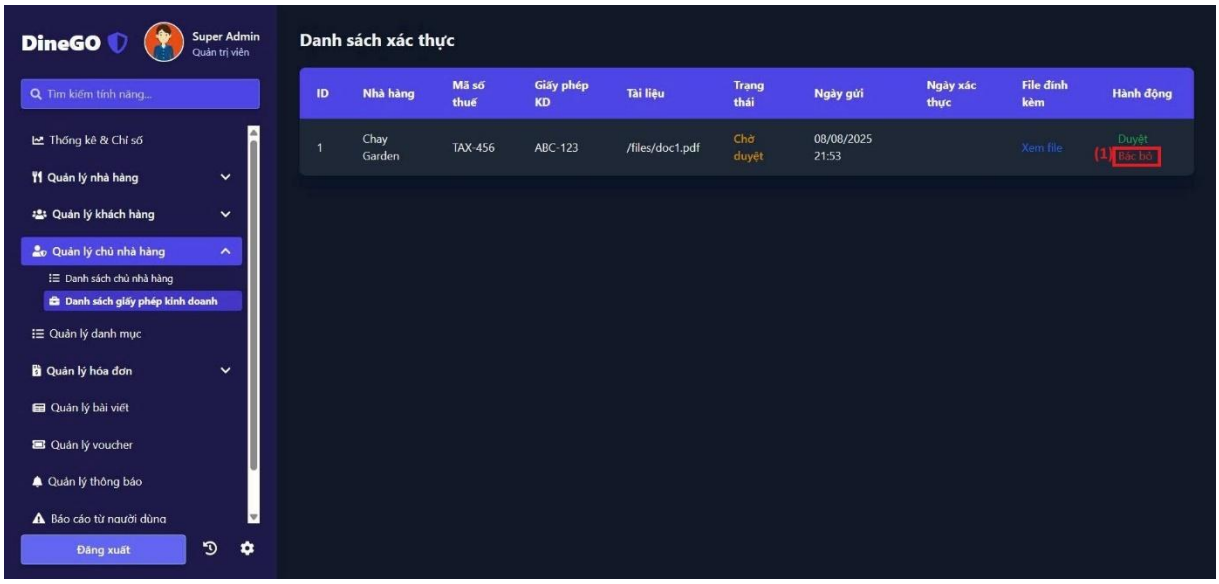


Figure 311: Business license page

This feature allows the Admin to reject a restaurant's business license request.

- (1) [Bác bỏ] button – Click to reject the selected business license.

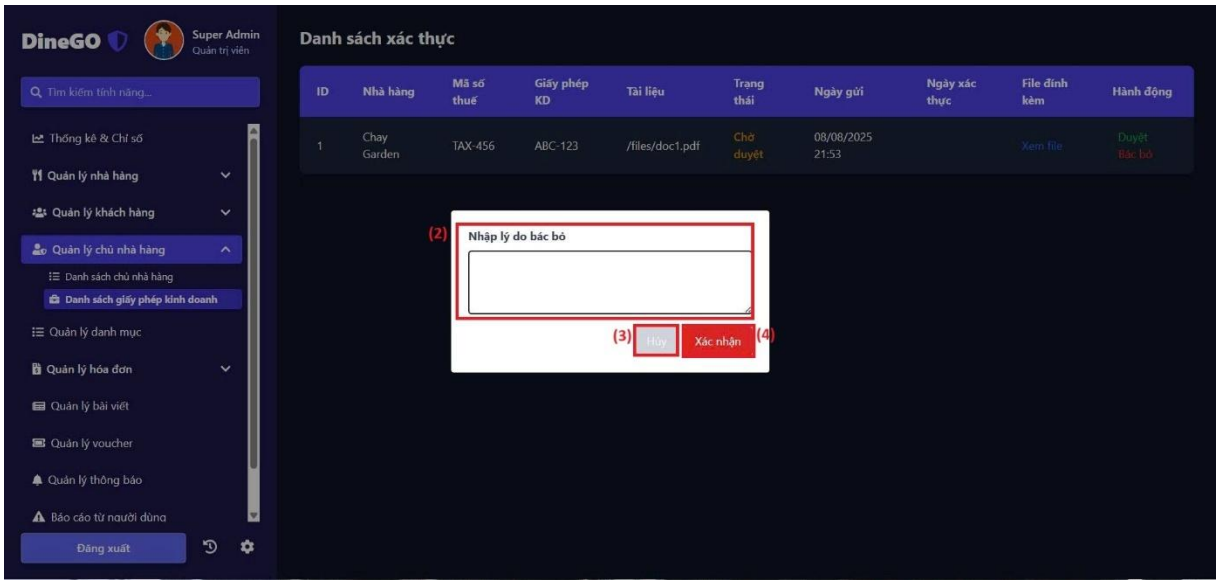


Figure 312: Reject business license page

- (2) Reason field – Enter the reason for rejection (required).
- (3) [Hủy] button – Cancel the rejection action.
- (4) [Xác nhận] button – Confirm and submit the rejection.

d. Manage restaurants

i. View restaurant

Figure 313: View restaurant page of DineGO

This feature allows the Admin to view and manage the list of all restaurants

- (1) [Danh sách nhà hàng]tab to access the list of restaurants.
- (2) Displays detail of each restaurant
    - The list should display all restaurants with complete detail.

ii. Create restaurant

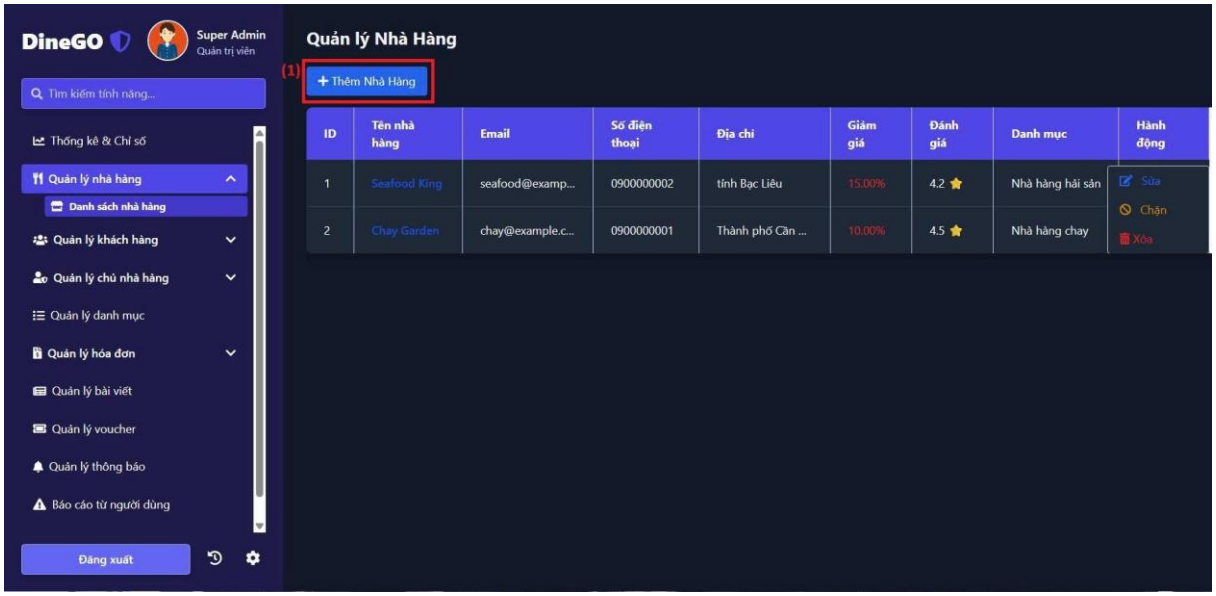


Figure 314: View restaurant page of DineGO

This feature allows the Admin to add a new restaurant to the system.

- (1) [Thêm Nhà Hàng] button to open the add restaurant form.

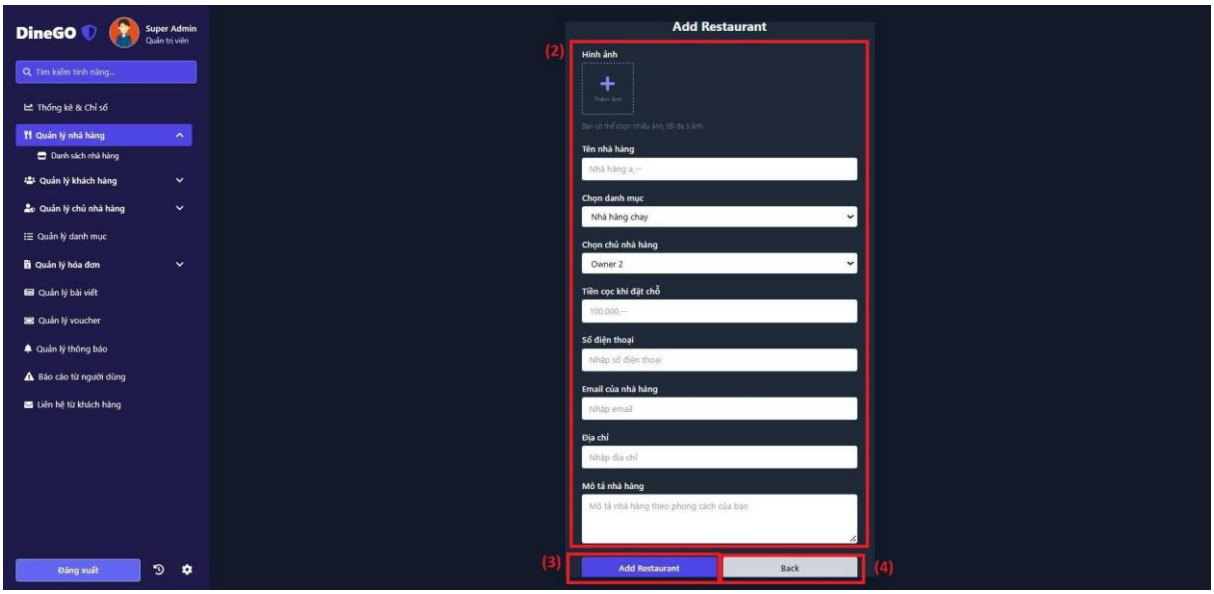


Figure 315: Create restaurant of DineGO

- (2) Fill all fields
- (3) [Thêm nhà hàng] button to save the new restaurant.
- (4) [Quay lại] button to return to the restaurant list without saving.
    - "Tên nhà hàng" is required and must be unique within owners.
    - "Danh mục" and "Chủ nhà hàng" are required selections.
    - "Tiền cọc khi đặt chỗ" must be a number ≥ 0.
    - "Số điện thoại" must be a valid format
    - "Email" must be in a valid email format.
    - On success, the restaurant appears in the list; on failure, show an error message.

iii. Block restaurant

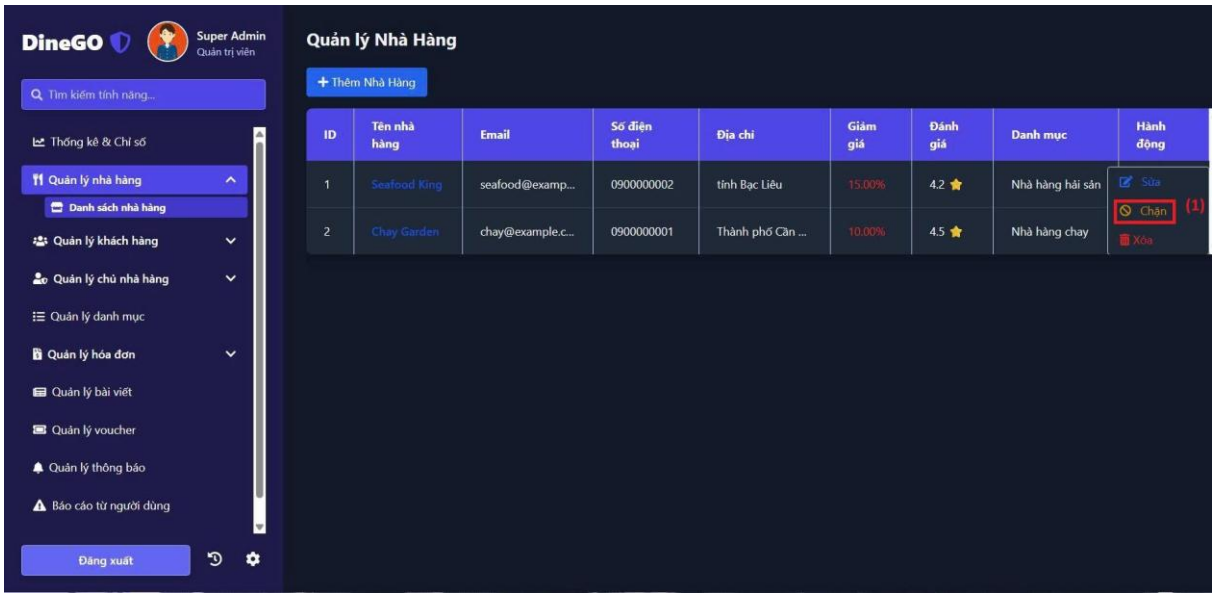


Figure 316: View restaurant page of DineGO

This feature allows the Admin to block a restaurant from the system.

- (1) [Chặn] icon to open the block restaurant form.

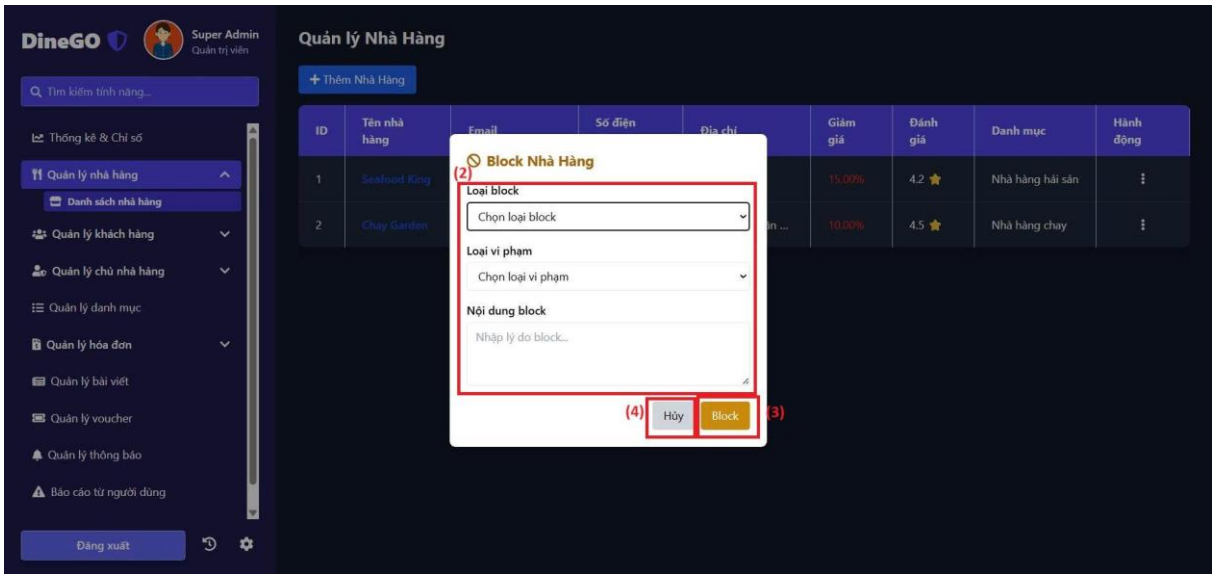


Figure 317: Block restaurant page of DineGO

- (2) Fill and select all field block form
- (3) [Block] button to confirm and save the block action.
- (4) [Hủy] button to cancel and close the form without blocking.
    - "Loại block" is required.
    - "Loại vi phạm" is required.
    - "Nội dung block" is required and must be at least 5 characters.
    - If blocking is successful, the restaurant's status changes to Blocked in the list.
    - If blocking fails, an error message is displayed.

iv. Update restaurant

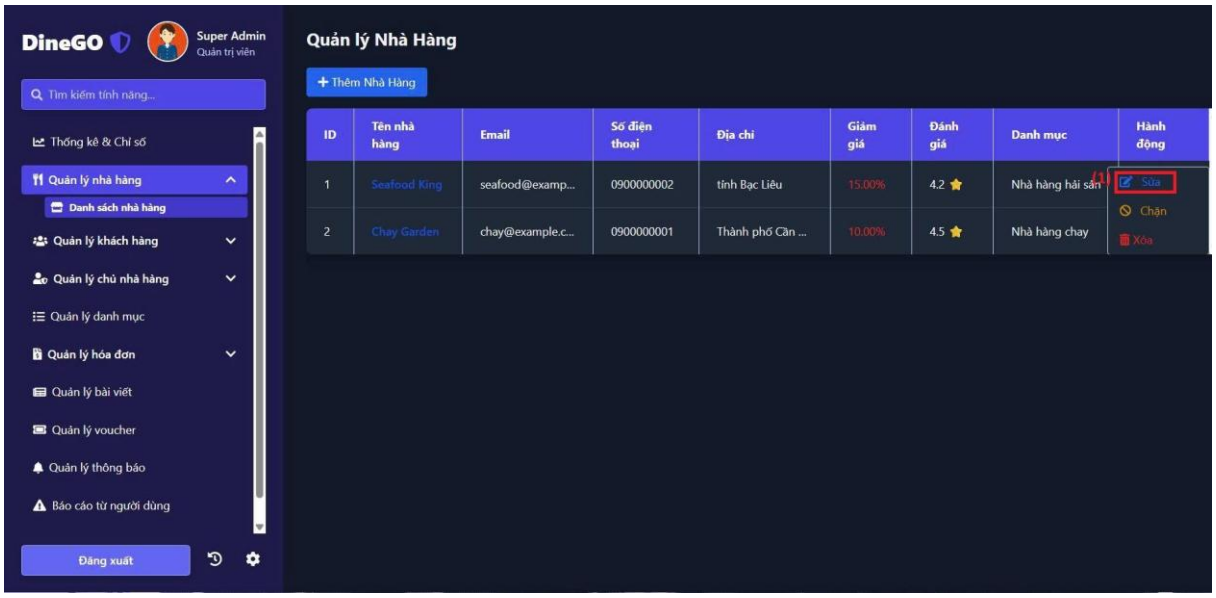


Figure 318: View restaurant page of DineGO

This feature allows the Admin to edit an existing restaurant's detail.

- (1) [Sửa] icon in the Hành động column of the restaurant list to open the edit restaurant form.

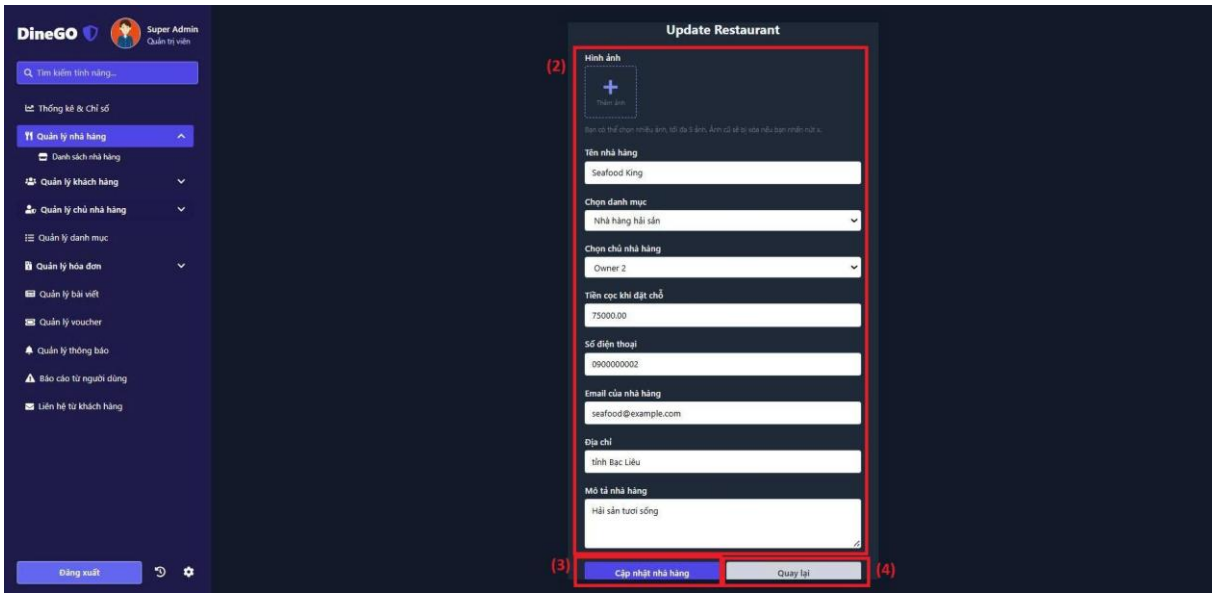


Figure 319: Update restaurant form of DineGO

- (2) Enter the updated restaurant information
- (3) [Cập nhật nhà hàng] button to save the changes.
- (4) [Quay lại] button to return to the restaurant list without saving.
    - “Tên nhà hàng” is required and must be unique within the same owner.
    - “Danh mục” and “Chủ nhà hang” are required selections.
    - “Tiền cọc” must be a number ≥ 0.
    - “Số điện thoại” must be valid
    - “Email" must be in valid email format.
    - Images accepted formats; each within the configured size limit.
    - If the update is successful, the restaurant detail are updated in the list.
    - If the update fails, an error message is displayed.

v. Delete restaurant

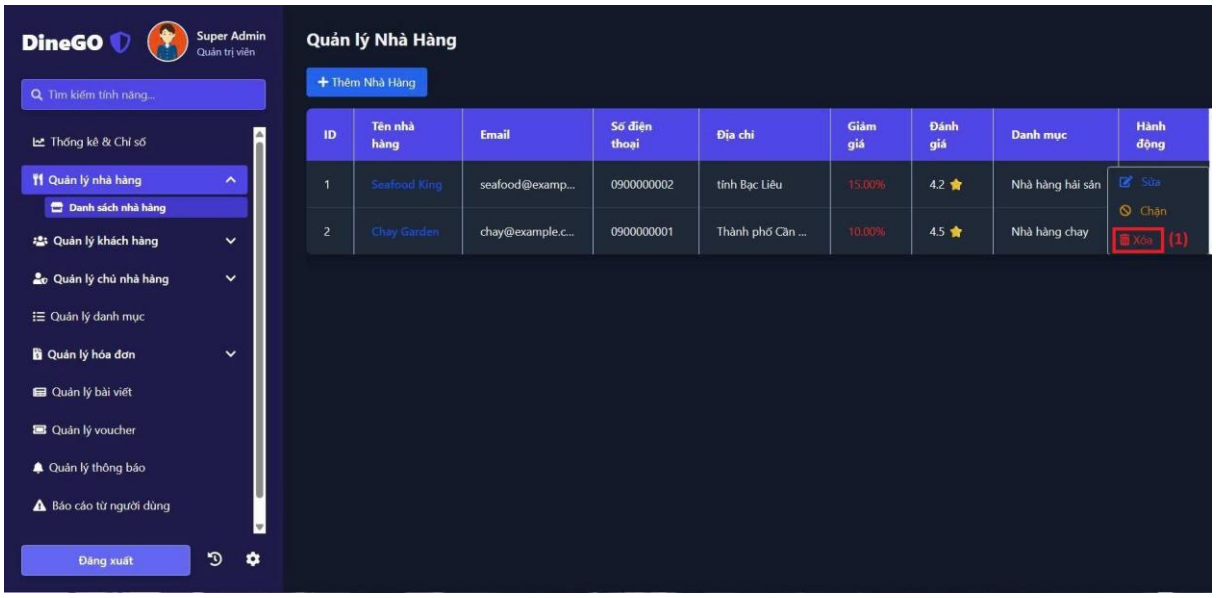


Figure 320: View restaurant page of DineGO

This feature allows the Admin to remove an existing restaurant

- (1) [Xóa] icon to open the deletion confirmation dialog.

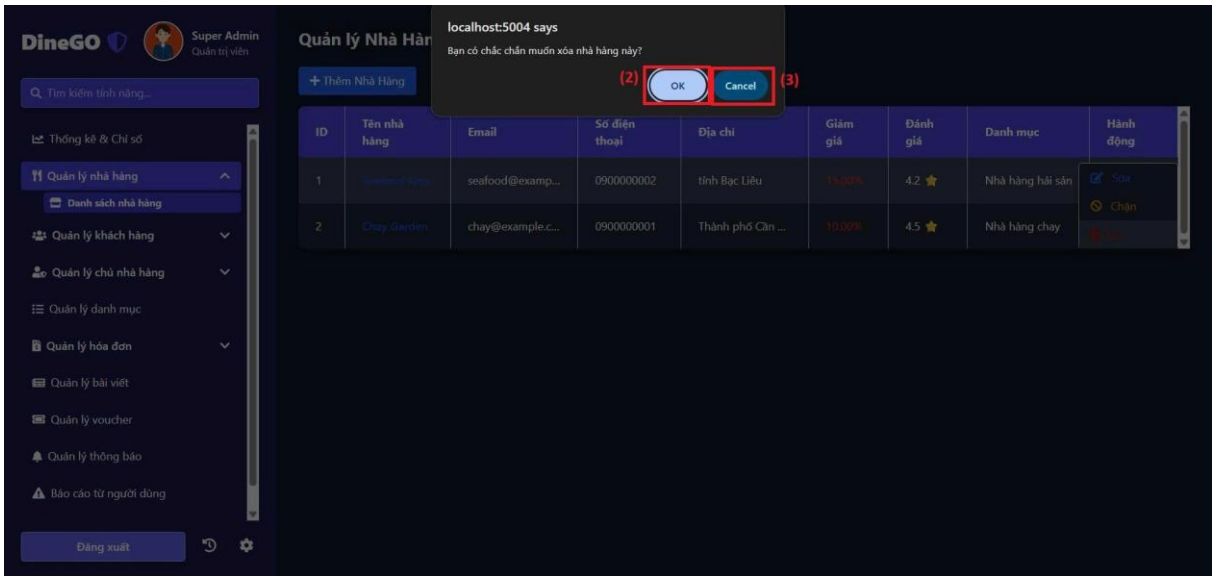


Figure 321: Confirm delete restaurant page

(2) [OK] button to confirm deletion.
(3) [Cancel] button to abort deletion and keep the restaurant.

- A confirmation prompt must appear before deletion.
- Only restaurants without active bookings or unpaid bills can be deleted.
- Deleted restaurants are removed from the list immediately.
- If deletion is successful, the restaurant disappears from the list.
- If deletion fails, an error message is displayed.

e. Manage restaurant categories

i. View category

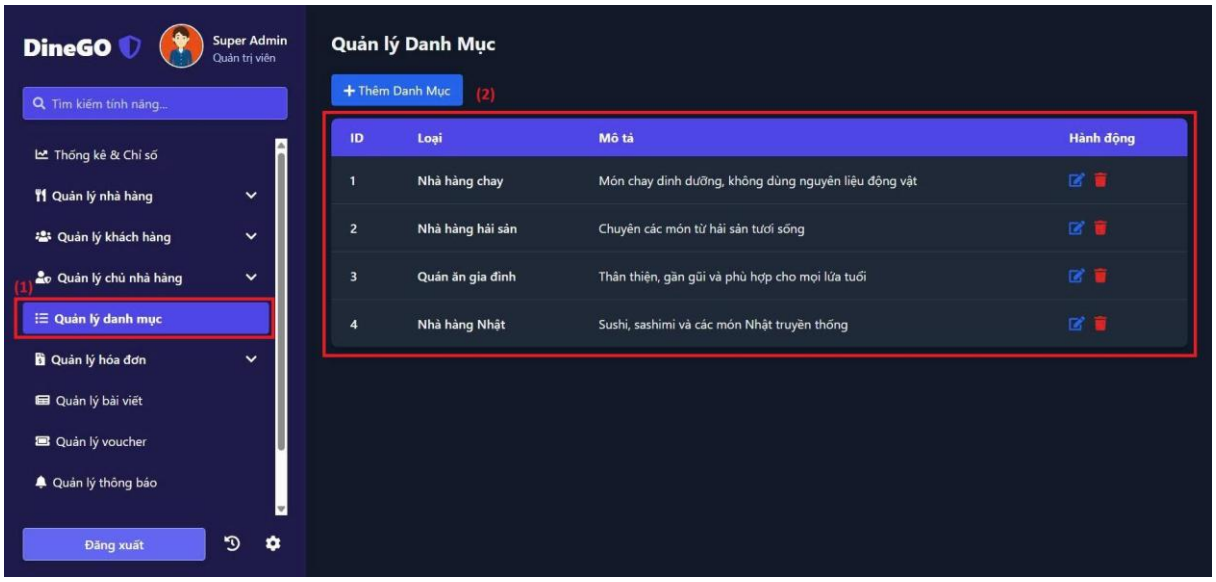


Figure 322: View category list of DineGO

This feature allows the Super Admin to view and manage all restaurant categories in the system.

- (1) [Quản lí danh mục] tab to open the category management page
- (2) View all category in list
    - The table must load all categories from the database accurately.
    - Category names and descriptions must match the stored data.

ii. Create category

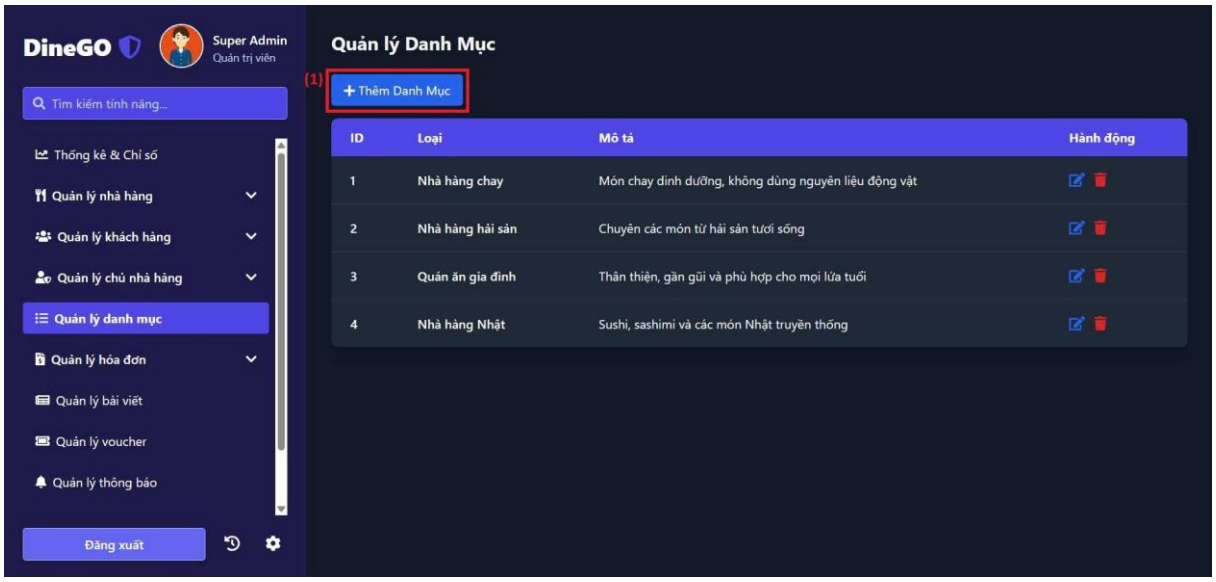


Figure 323: Category page of DineGO

This feature allows the Admin to create a new restaurant category.

- (1) [Thêm danh mục] button to open create category form

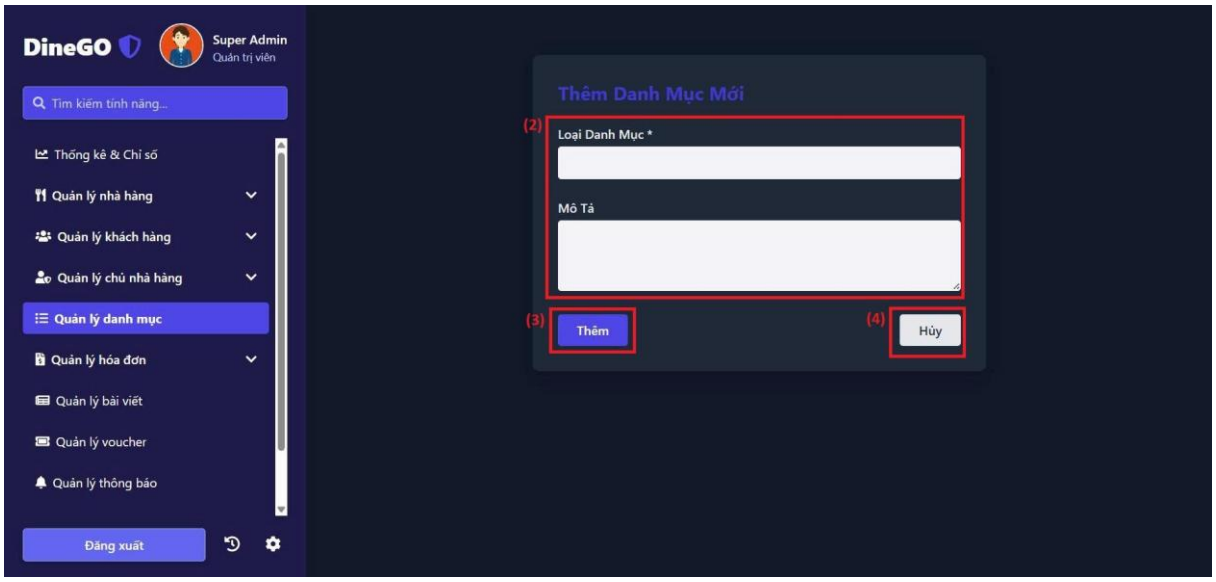


Figure 324: Create category page of DineGO

- (2) Fill in category detail
- (3) [Thêm] button to add category
- (4) [Hủy] button to cancel operation
    - "Loại danh mục" is mandatory and must not duplicate existing categories.
    - "Mô tả" can be left empty
    - Upon successful creation, the new category should appear in the category list without requiring a full page reload.

iii. Update category

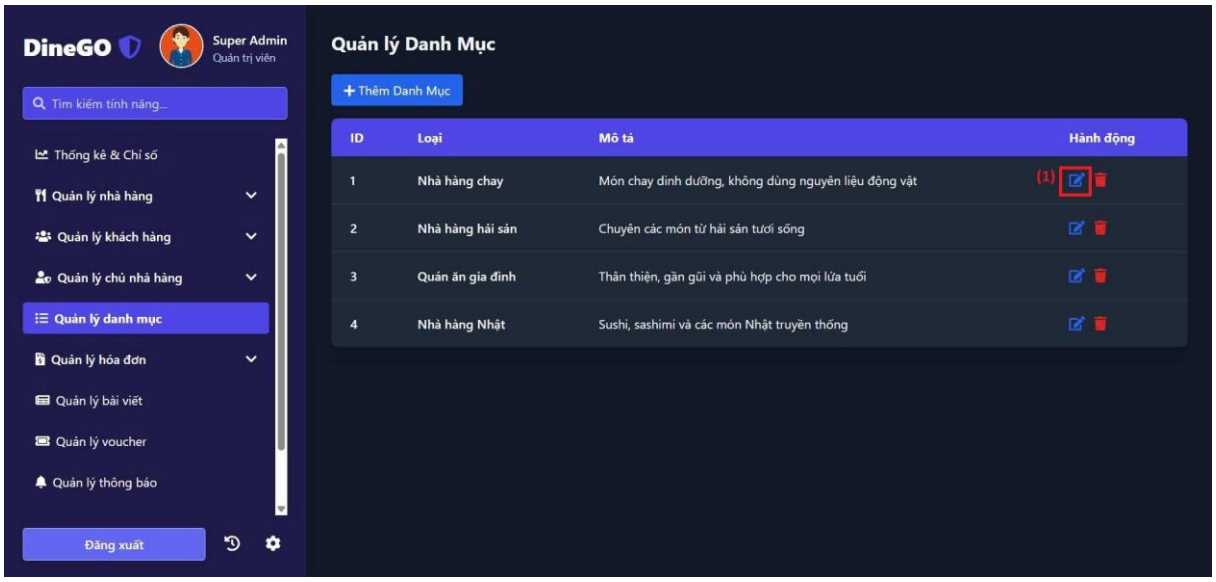


Figure 325: Category page in DineGO

This feature allows the Admin to modify an existing restaurant category's detail in the system.

- (1) [edit icon] next to the category you want to modify

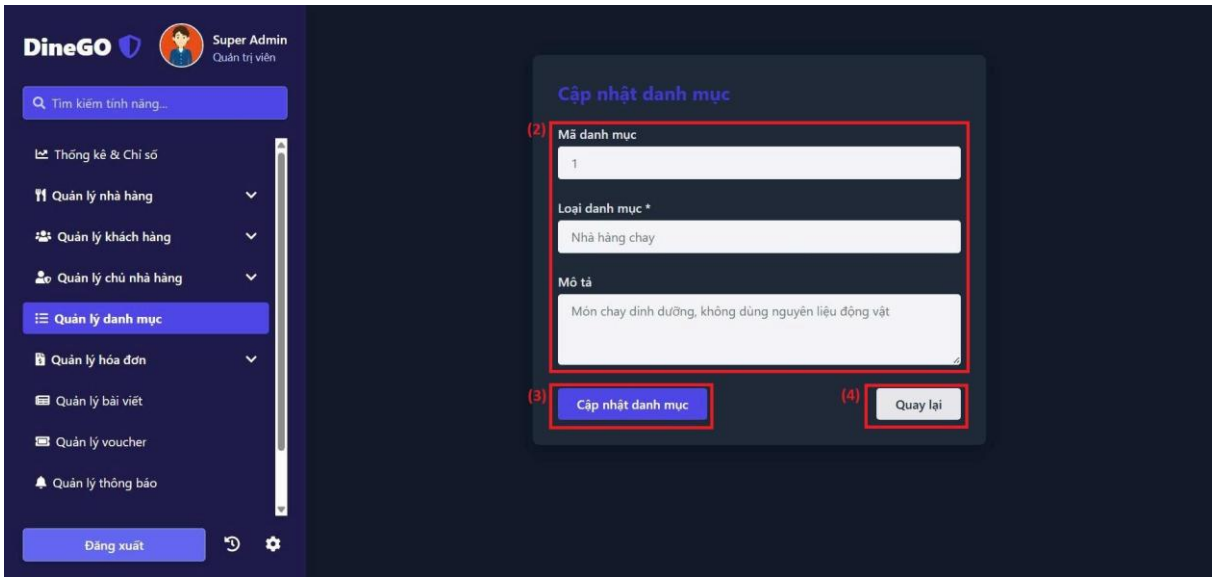


Figure 326: Update category in DineGO

- (2) Enter information you want to update
- (3) [Cập nhật danh mục] button to save the changes.
- (4) [Quay lại] button to return to the category list without saving.
    - All required fields must be filled in before updating.
    - "Loại danh mục" must not be empty or duplicated.
    - On successful update, the system should refresh the category list with updated detail.
    - If the update fails, an error message should be displayed.

iv. Delete category

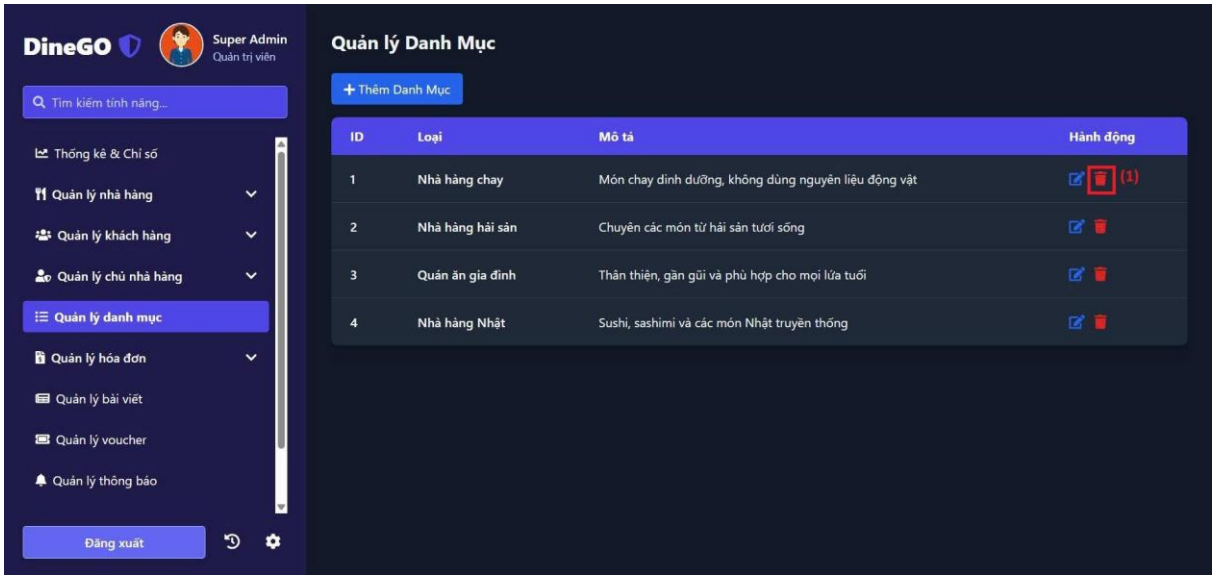


Figure 327: Category page in DineGO

This feature allows the Super Admin to delete a restaurant category from the system.

- (1) [Delete icon] next to the category you wish to remove.

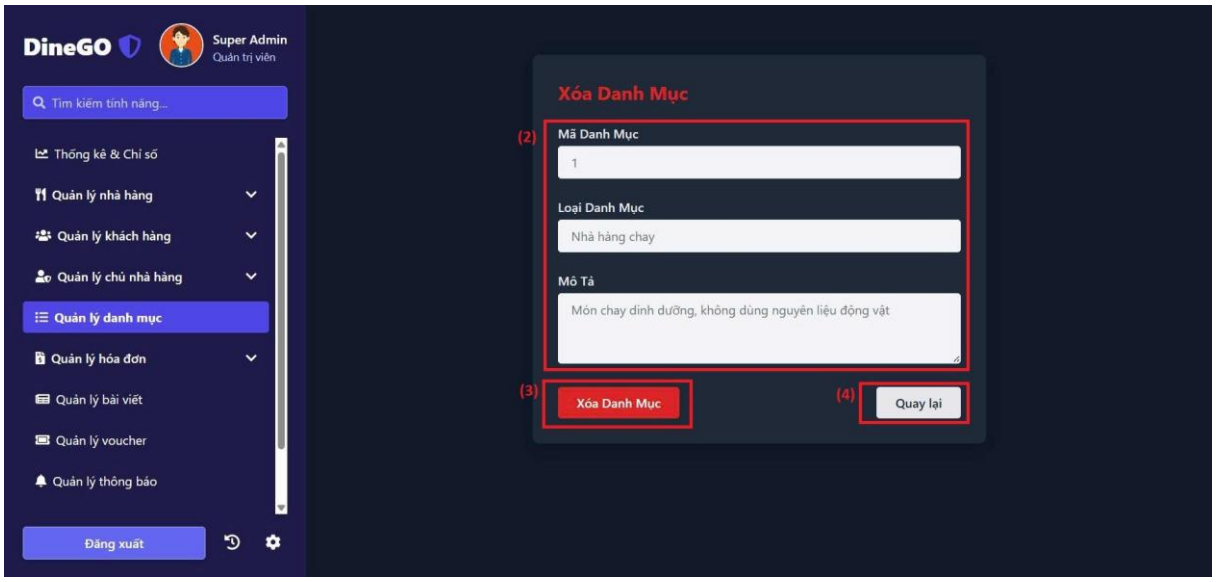


Figure 328: Delete category process in DineGO

- (2) View category detail before deletion
- (3) [Xóa Danh Mục] button to permanently remove the category from the database
- (4) [Quay lại] button to return to the category list without deleting.
    - The category detail displayed must match the database record.
    - Clicking [Xóa danh mục] should remove the category and refresh the category list.
    - A confirmation or success message should be displayed upon successful deletion.

f. Manage food orders

i. View all orders

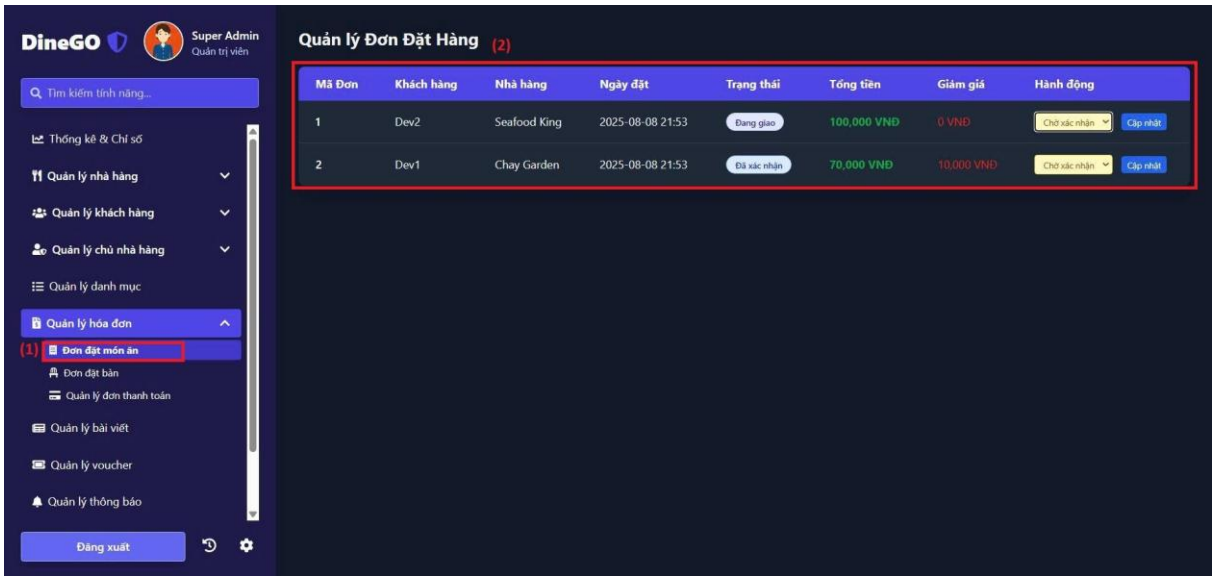


Figure 329: View order list of DineGO

This feature allows the Admin to view and manage all food orders

- (1) [Đơn đặt món ăn] menu item to open the order list page.
- (2) Order list table displaying
    - Status updates must follow the allowed status flow
    - "Tổng tiền" and "Giảm giá" must be correctly calculated from the order detail.
    - If the page loads successfully, all orders are listed with accurate detail.
    - If loading fails, an error message is displayed.

ii. Process all orders

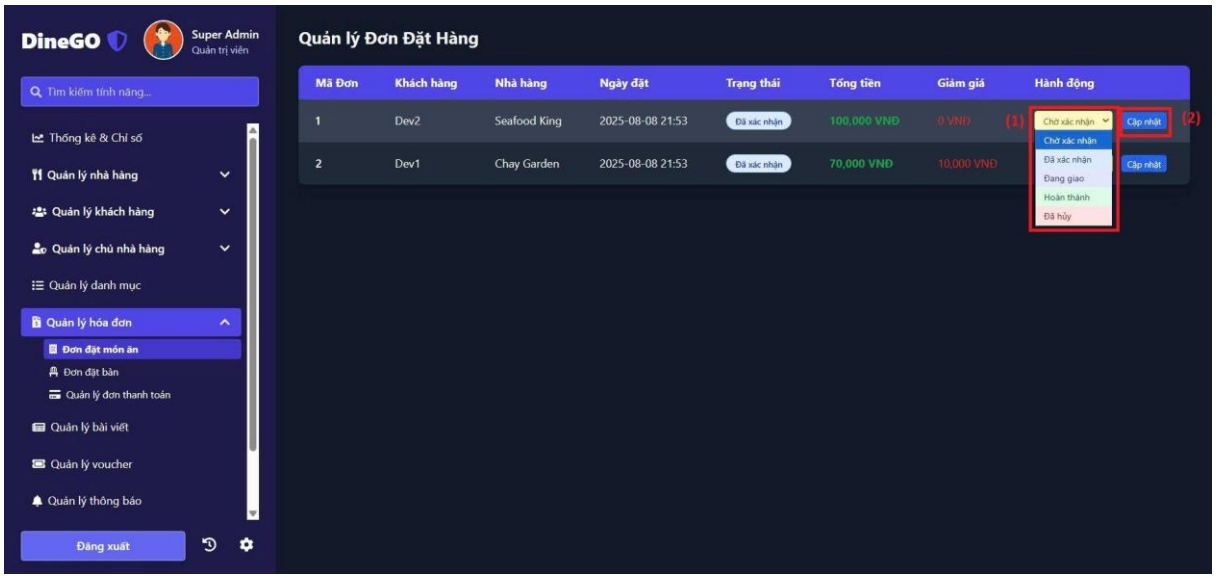


Figure 330: View order list of DineGO

This feature allows the Admin to change the status of an existing food order.

- (1) Status dropdown in the Hành động column of the order list to select a new order status
- (2) [Cập nhật] button to save the selected status.
    - Status changes must follow the predefined order flow
    - Once an order is "Hoàn thành" or "Đã hủy," it cannot be changed again.
    - If the update is successful, the new status appears immediately in the list.
    - If the update fails, an error message is displayed.

g. View table reservation

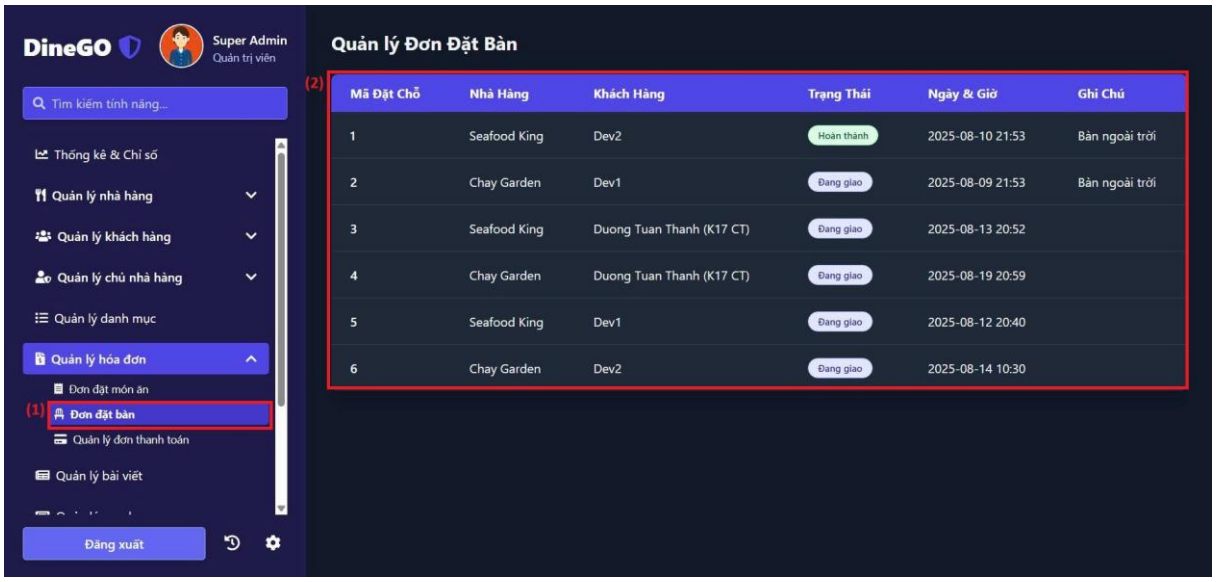


Figure 331: Table reservation in DineGO

This feature allows the Admin to view and manage all table reservations

- (1) [Đơn đặt bàn] menu to open the table reservation management page.
- (2) Displays detailed information for each reservation
    - Only Admin accounts can access this page.
    - "Trạng thái" must match the latest update from the restaurant.
    - "Ngày và giờ" must follow the format YYYY-MM-DD HH:mm

h. Manage payments

i. View payments

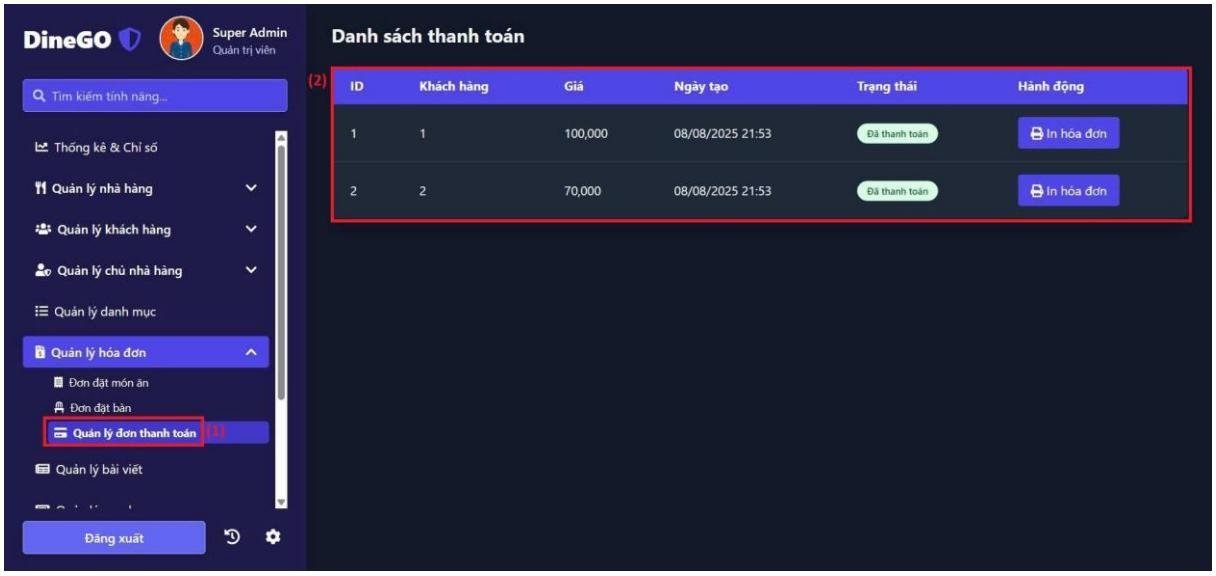


Figure 332: Payment manage page in DineGO

This feature allows the Admin to view the list of completed payment orders.

- (1) [Quản lý đơn thanh toán] – Opens the payment order management page.
- (2) Displays all payment orders
    - Only successfully paid orders will appear in this list.

- Payment detail must match the corresponding order in the system.
- "In hóa đơn" should generate accurate and complete information.

ii. Print invoice

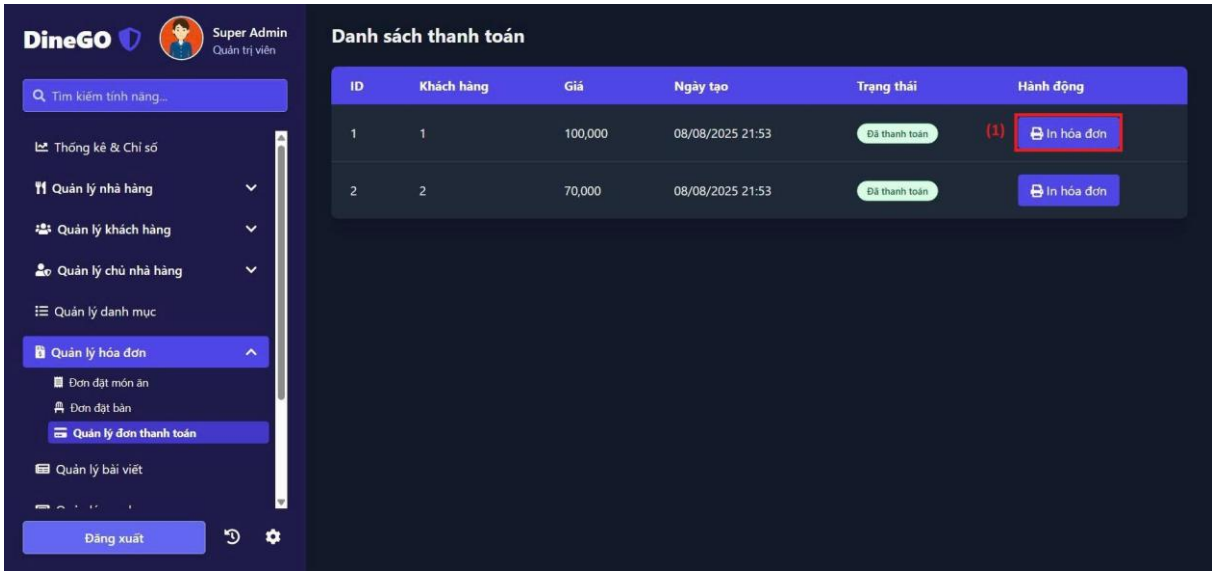


Figure 333: Payment manages page in DineGO

This feature allows the Admin to print the payment invoice for a completed order.

- (1) [In hoá đơn] button to open the invoice detail view.

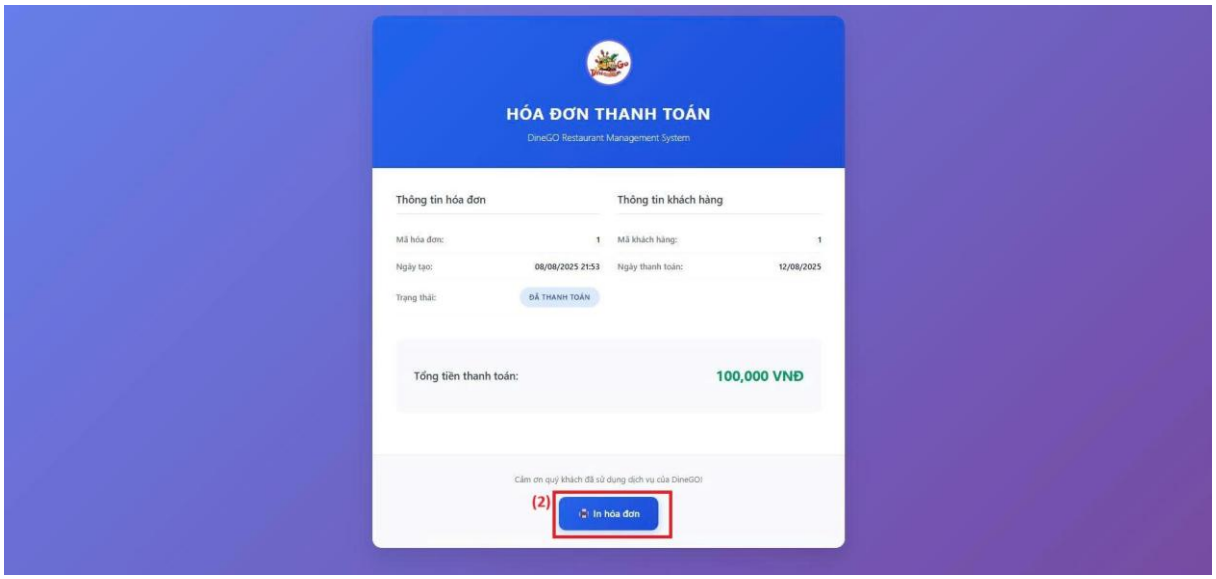


Figure 334: Print invoice in DineGO

- (2) [In hoá đơn] buttonto prints the displayed invoice
  - Invoice detail must match the payment record in the system.
  - The printed invoice should include all necessary detail: mã hóa đơn, thông tin khách hàng, ngày/giờ, số lượng, and trạng thái
  - Ensiure the printer is properly connected and configured for successful printing.

i. Mange blog

i. View blog

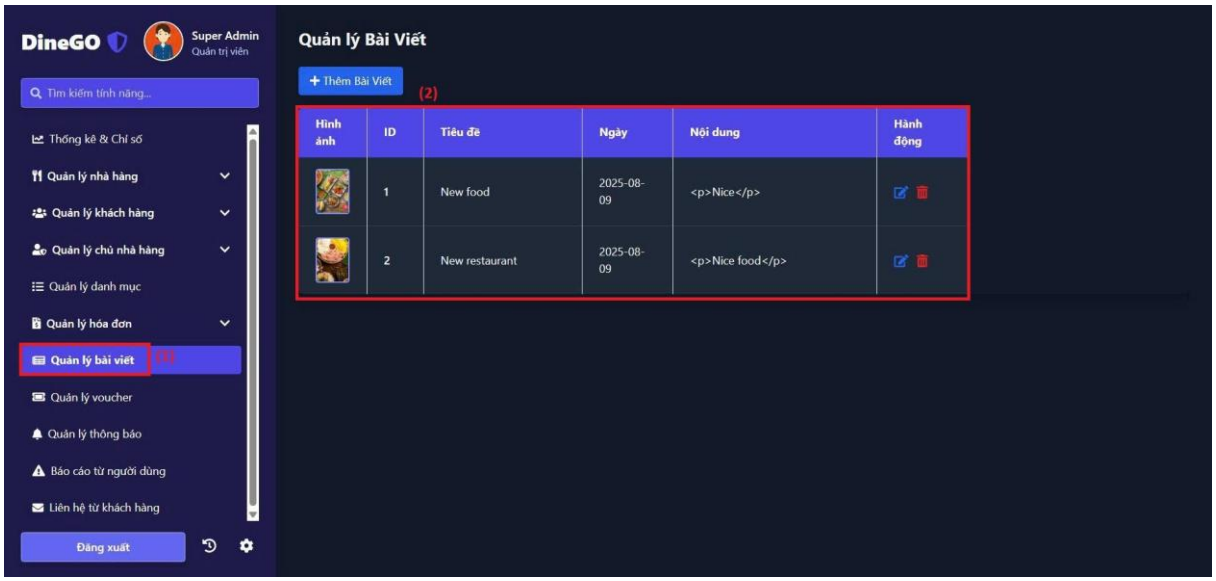


Figure 335: Blog manages page in DineGO

This feature allows the Admin to view and manage all blog posts in the system.

- (1) [Quản lý bài viết] menu item to open the blog post list page.
- (2) Blog post list table displaying
  - Images must load correctly from the stored file path.
  - "Tiêu đề" and "Nội dung" must display without broken HTML tags.
  - Posts should be listed in reverse chronological order
  - If the page loads successfully, all posts are displayed in the table with correct data.
  - If loading fails, an error message is shown.

ii. Create blog

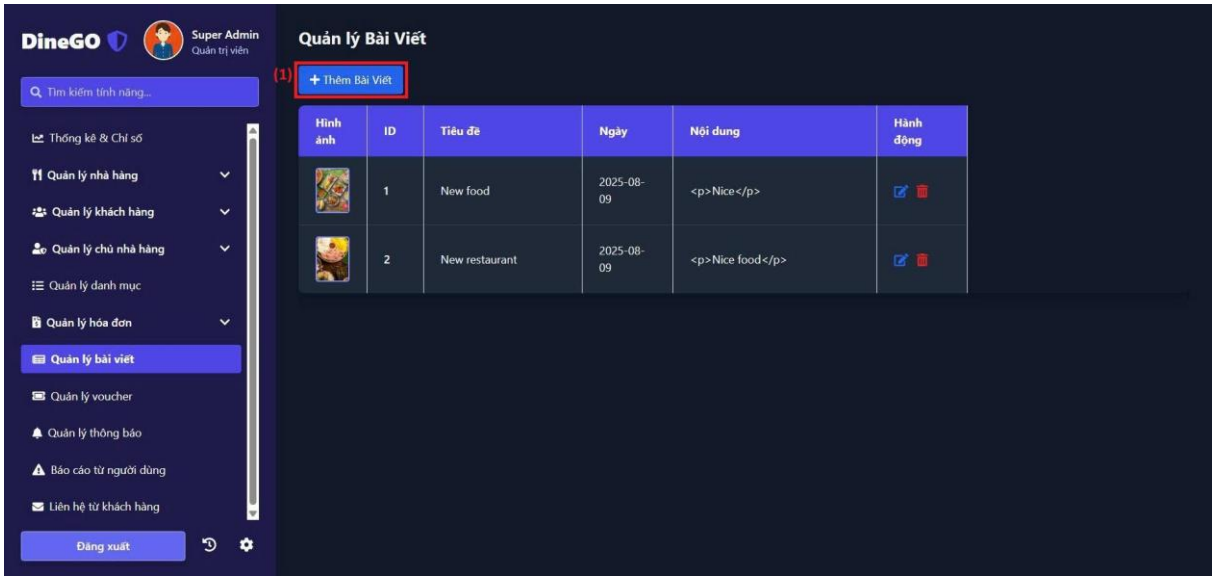


Figure 336: Blog manages page in DineGO

This feature allows the Admin to create a new blog post in the system.

- (1) [Thêm Bài Viết] button to open the add blog post form.

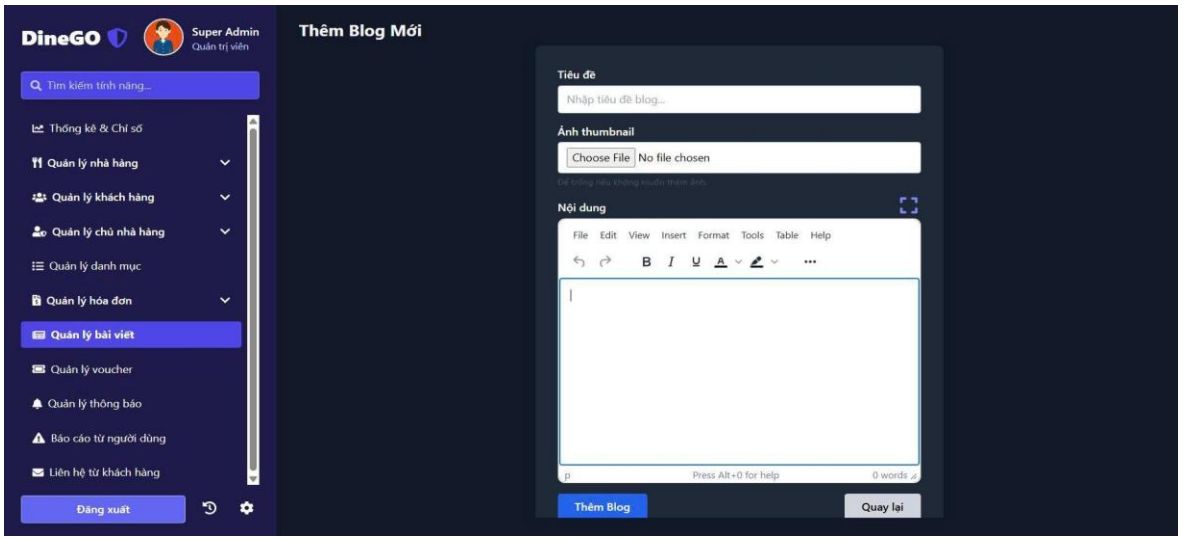


Figure 337: Add blog form in DineGO

- (2) Fill all fields
- (3) [Thêm Blog] button to save the new blog post.
- (4) [Quay lại] button to return to the blog post list without saving.
    - "Tiêu đề" is required and must be unique.
    - "Nội dung" is required and should not be empty.
    - Uploaded thumbnail must be in JPG/PNG format and within the allowed size.
    - If the creation is successful, the new blog post appears in the list.
    - If creation fails, an error message is displayed.

iii. Update blog

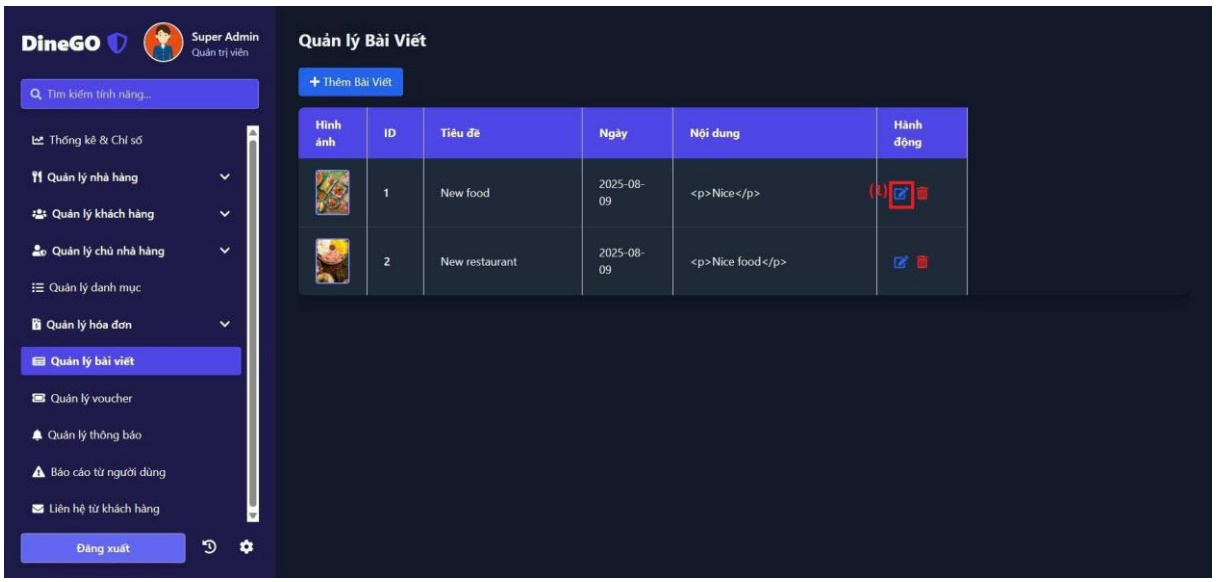


Figure 338: Blog manages page in DineGO

This feature allows the Admin to update an existing blog post

- (1) [Sửa] icon in the Hành động column to open the edit blog post form.

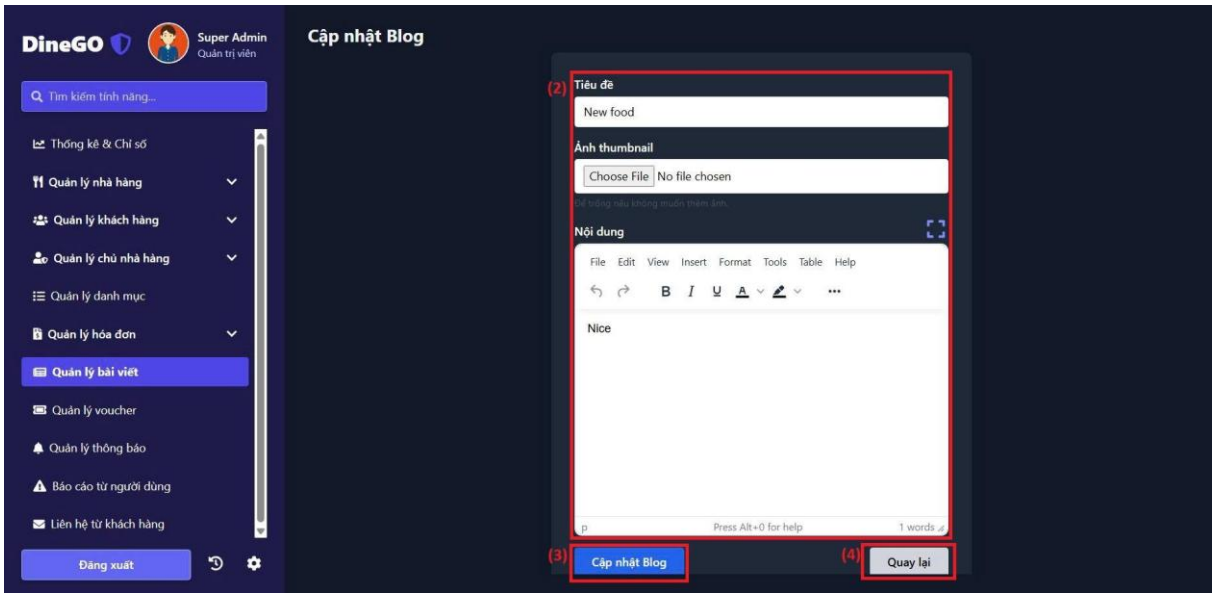


Figure 339: Edit blog form in DineGO

- (2) Enter the updated blog information to want update
- (3) [Cập nhật Blog] button to save the changes.
- (4) [Quay lại] button to return to the blog post list without saving.
    - "Tiêu đề" is required and must be unique.
    - "Nội dung" is required and should not be empty
    - Uploaded thumbnails must be in JPG/PNG format and within the allowed size.
    - If the update is successful, the changes are reflected in the blog post list.
    - If the update fails, an error message is displayed.

iv. Delete blog

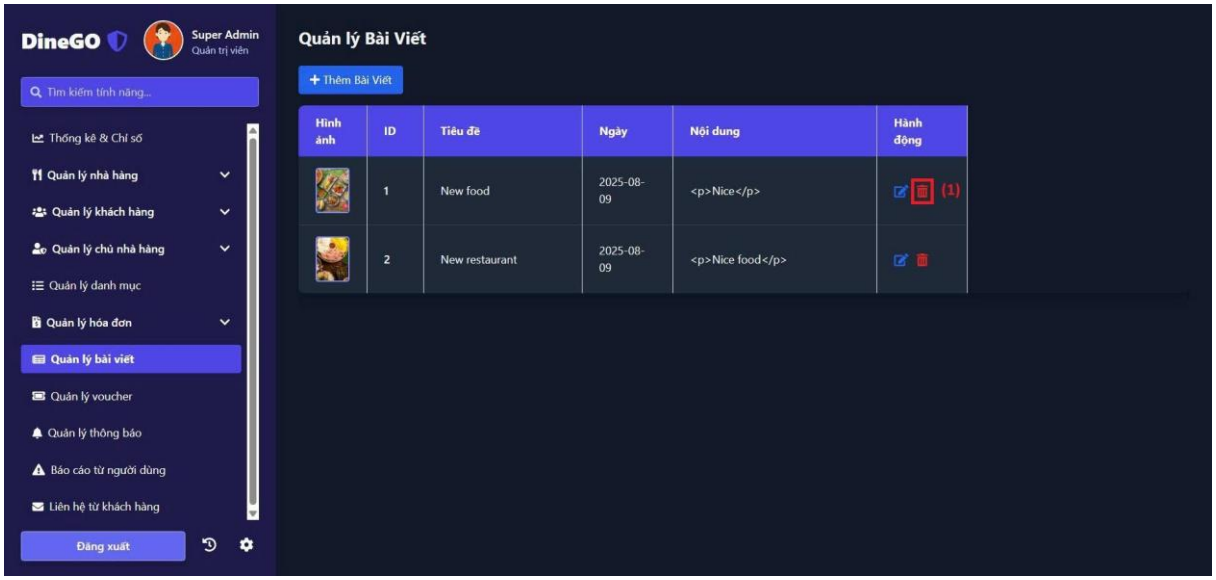


Figure 340: Blog manage page in DineGO

This feature allows the Admin to delete an existing blog post

- (1) [Delete] button to initiate deletion of the selected blog post.

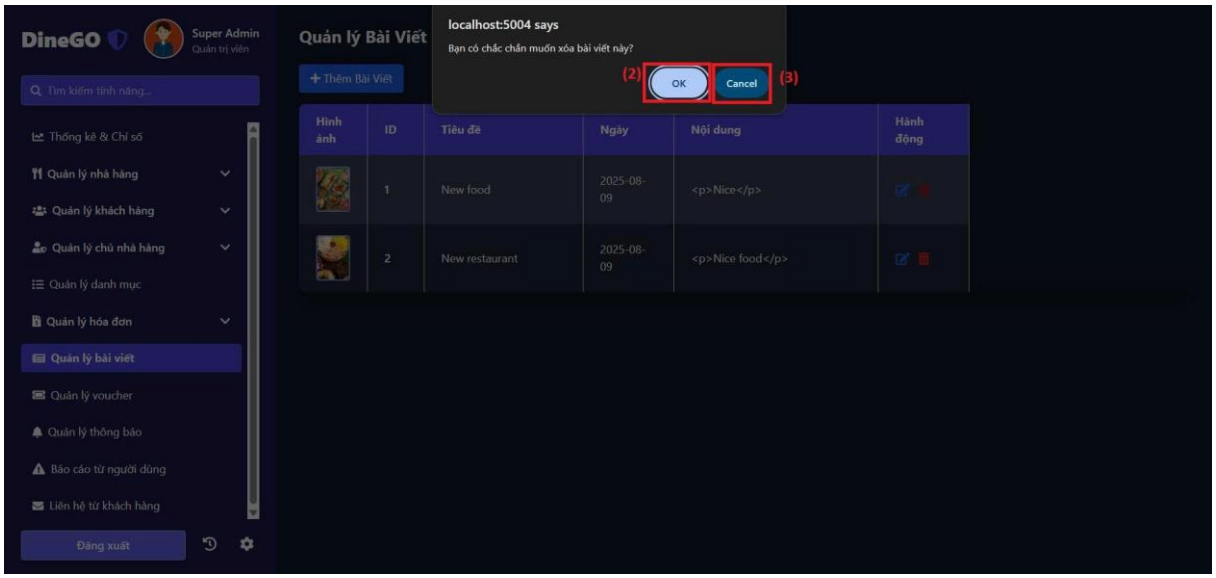


Figure 341: Delete confirm blog in DineGO

- (2) [OK] button in the confirmation dialog to proceed with deletion.
- (3) [Cancel] button in the confirmation dialog to abort the deletion process.
    - Deletion action requires user confirmation via a dialog box.
    - Once confirmed, the blog post is permanently removed from the list.
    - If cancelled, no changes are made and the blog post remains in the list.
    - After successful deletion, the list refreshes to reflect the removal.
    - If deletion fails, an error message is displayed.

j. Manage vouchers

i. View voucher

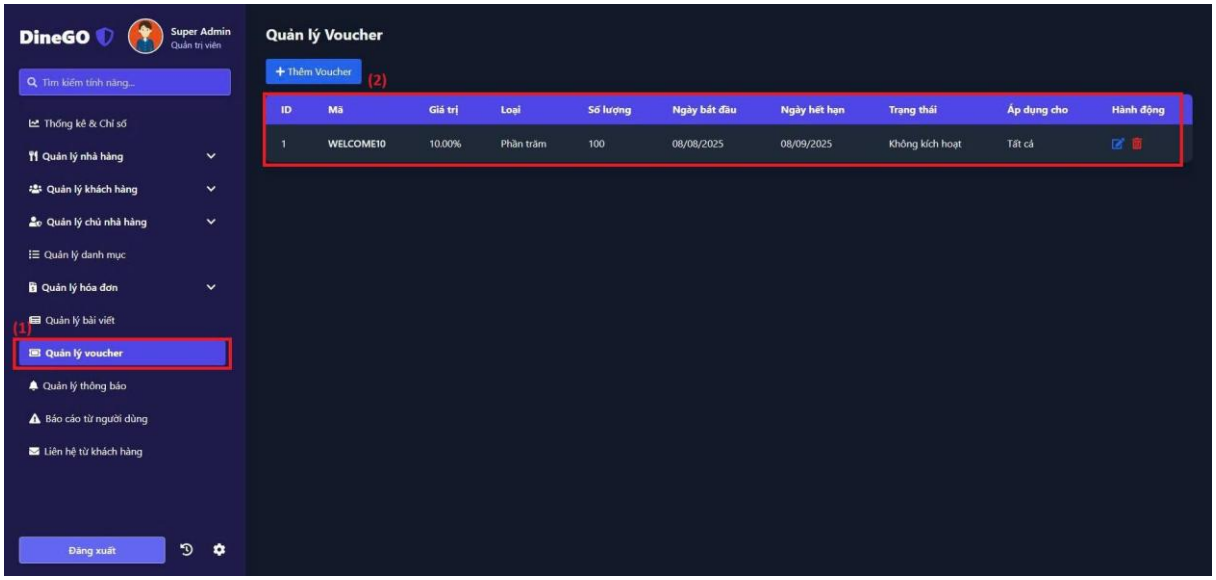


Figure 342: Voucher page in DineGO

This feature allows the Admin to view the list of vouchers

- (1) [Quản lý voucher] tab from the left navigation menu.
- (2) The table displays all vouchers with detail
    - All voucher information must match the database records.
    - Clicking edit or delete should navigate to the corresponding function.

ii. Create voucher

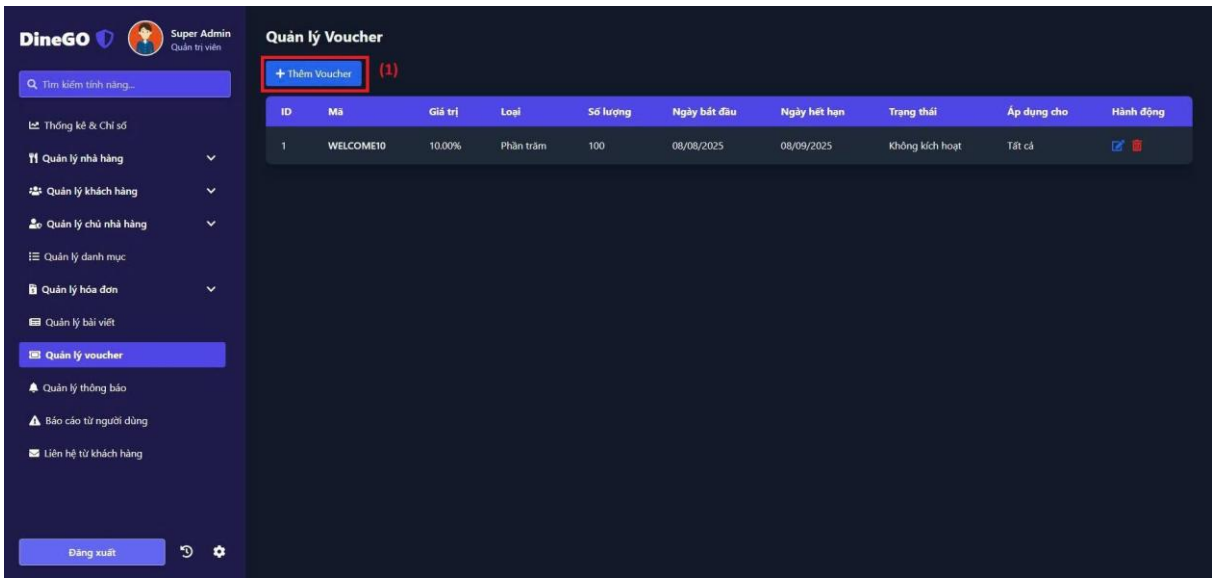


Figure 343: Voucher page in DineGO

This feature allows the Admin to create a new voucher

- (1) Click the [+ Thêm Voucher] button to open the "Thêm Voucher Mới" form.

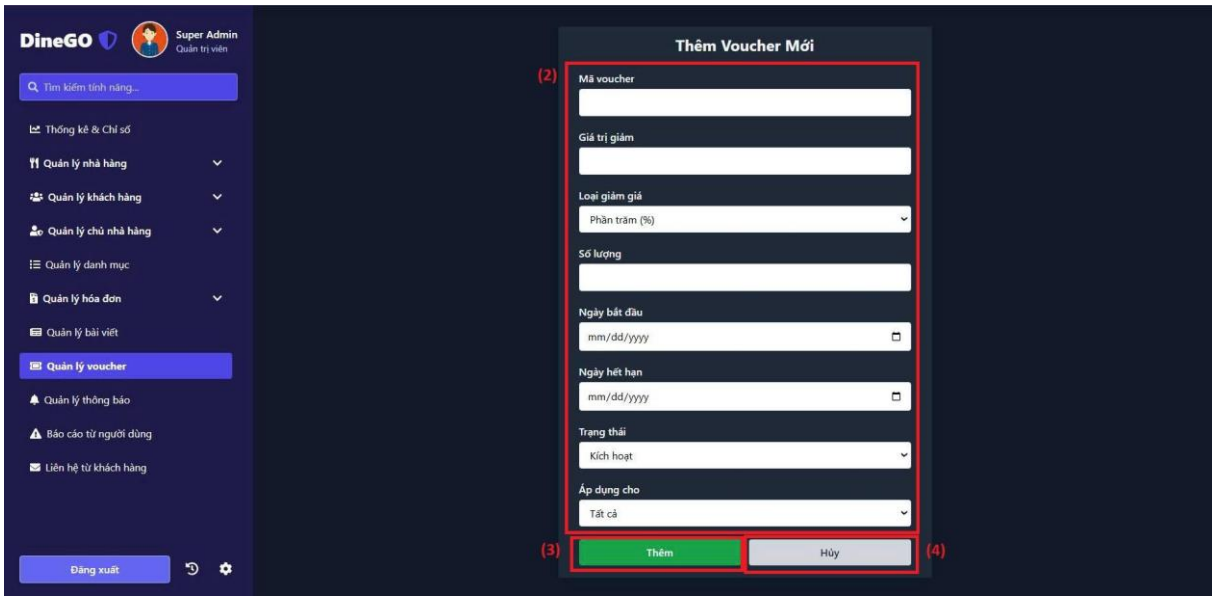


Figure 344: Create voucher in DineGO

- (2) Enter all required voucher detail

- (3) [Thêm] button to save the new voucher.
- (4) [Hủy] button to cancel and return to the voucher list without saving.
    - All required fields must be filled in before creating the voucher.
    - "Mã voucher" must be unique.
    - The start date must be earlier than the end date.

iii. Update voucher

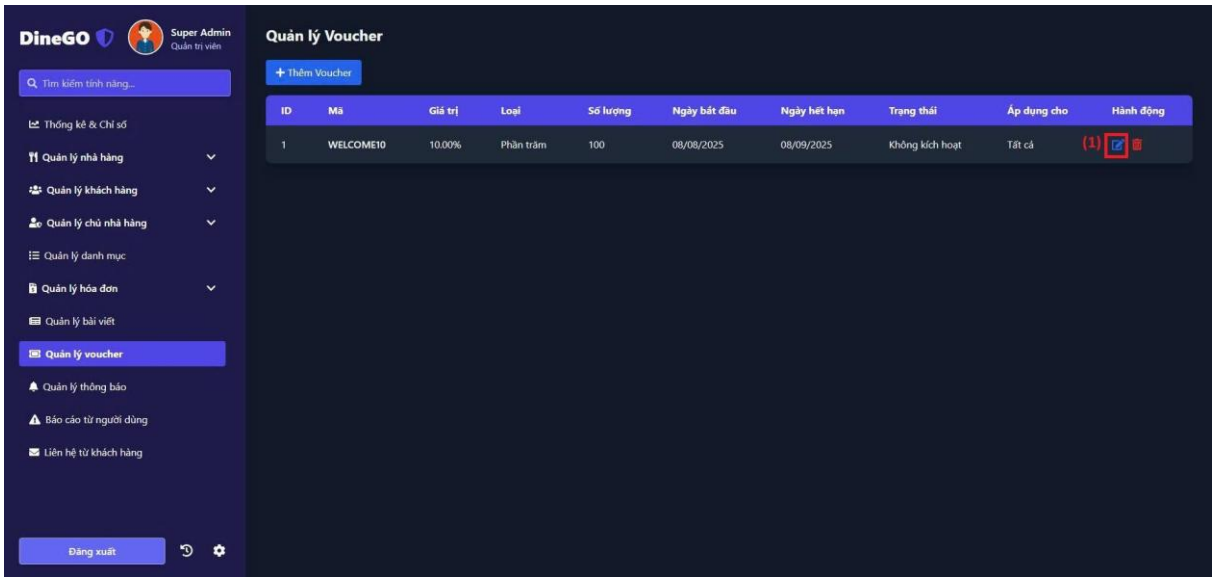


Figure 345: Voucher page in DineGO

This feature allows the Admin to update an existing voucher's detail

- (1) [edit icon] next to the voucher you want to modify.

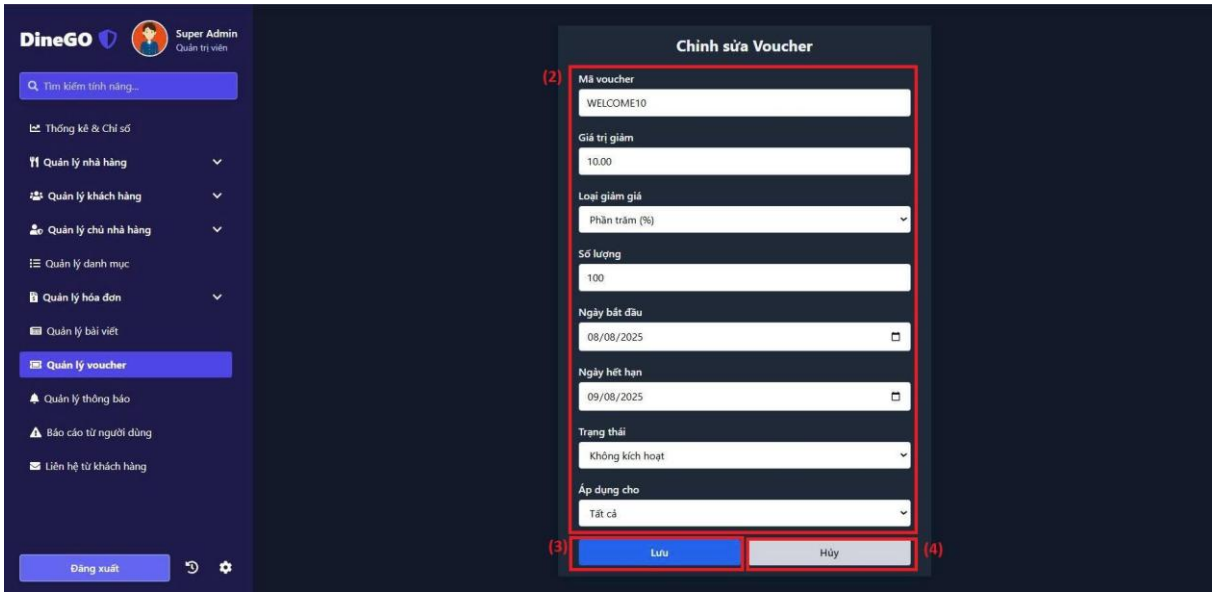


Figure 346: Update voucher in DineGO

- (2) Enter the updated voucher information.
- (3) [Lưu] button to save changes.
- (4) [Hủy] button to return without saving.

- All required fields must be completed before saving.
- “Mã voucher” must be unique and not empty.
- On success, the voucher list will refresh with updated detail.
- If the update fails, an error message will be displayed.

iv. Delete voucher

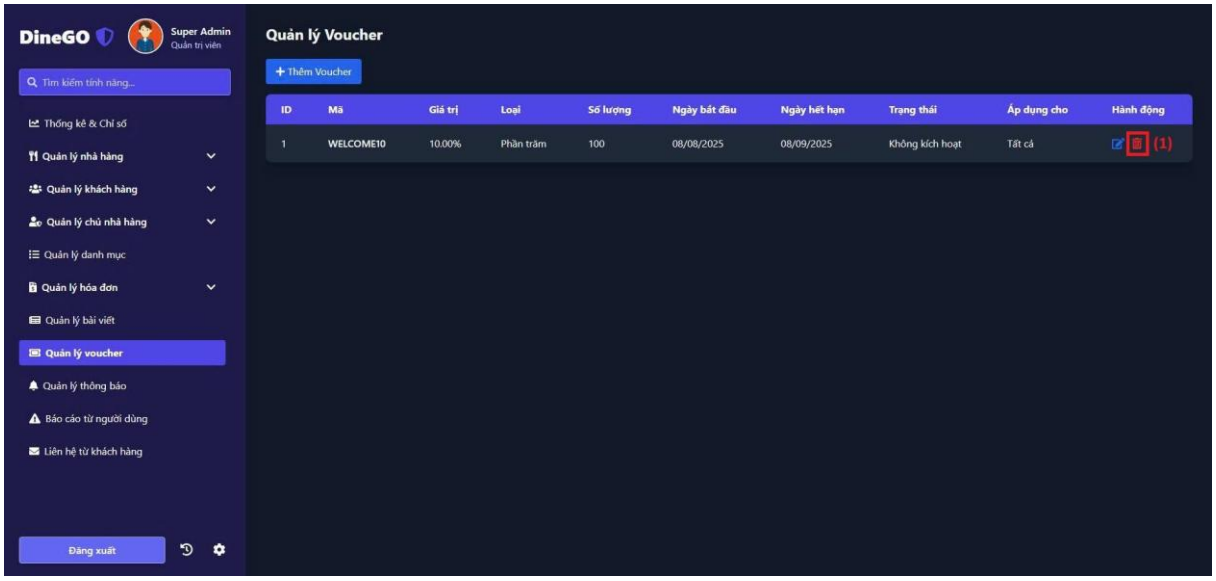


Figure 347: Voucher page in DineGO

This feature allows the Admin to remove an existing voucher

- (1) [Delete icon] next to the voucher you want to remove.

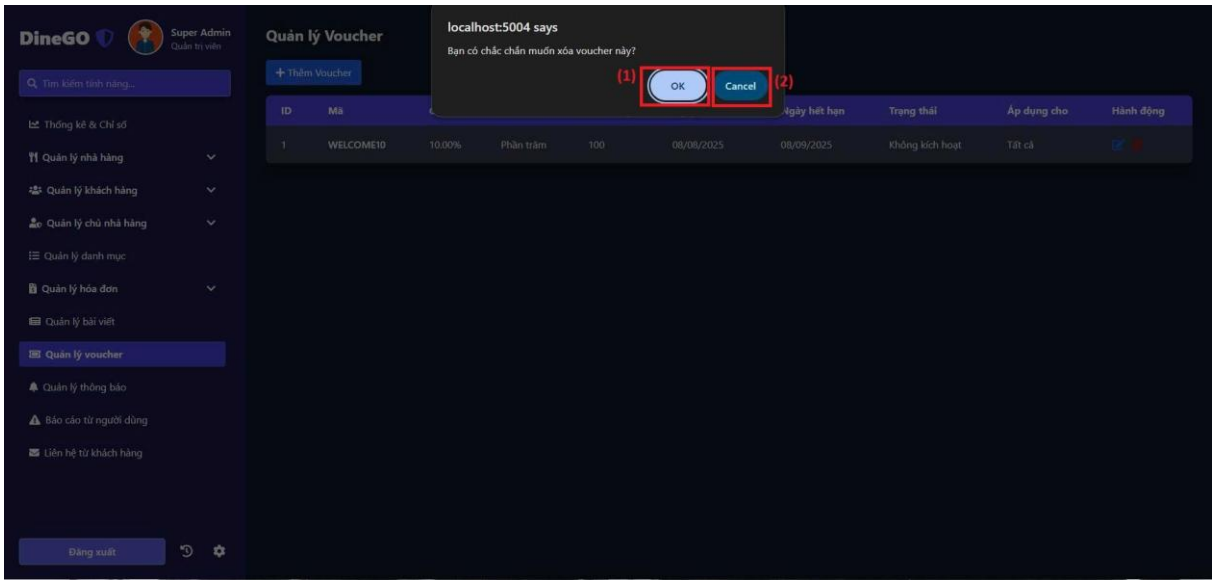


Figure 348: Delete voucher in DineGO

- (2) [OK] button to confirm deletion.
- (3) [Cancel] button to abort deletion.
  - If deletion is successful, the voucher is removed from the list.
  - If deletion fails, an error message should be displayed.

k. Manage notifications

i. View notification

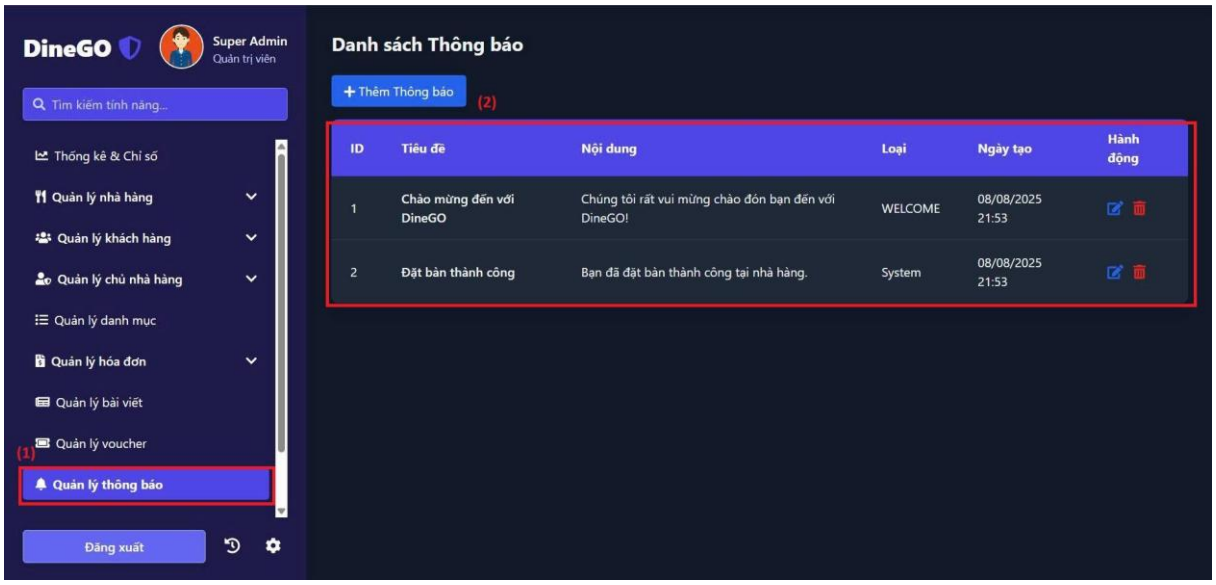


Figure 349: Notification page in DineGO

This feature allows the Admin to manage and view all system notifications.

- (1) [Quản lý thông báo] menu item to open the notification list page.
- (2) Table listing notifications
  - Admins can review, edit, or remove notifications from the system.

ii. Create notification

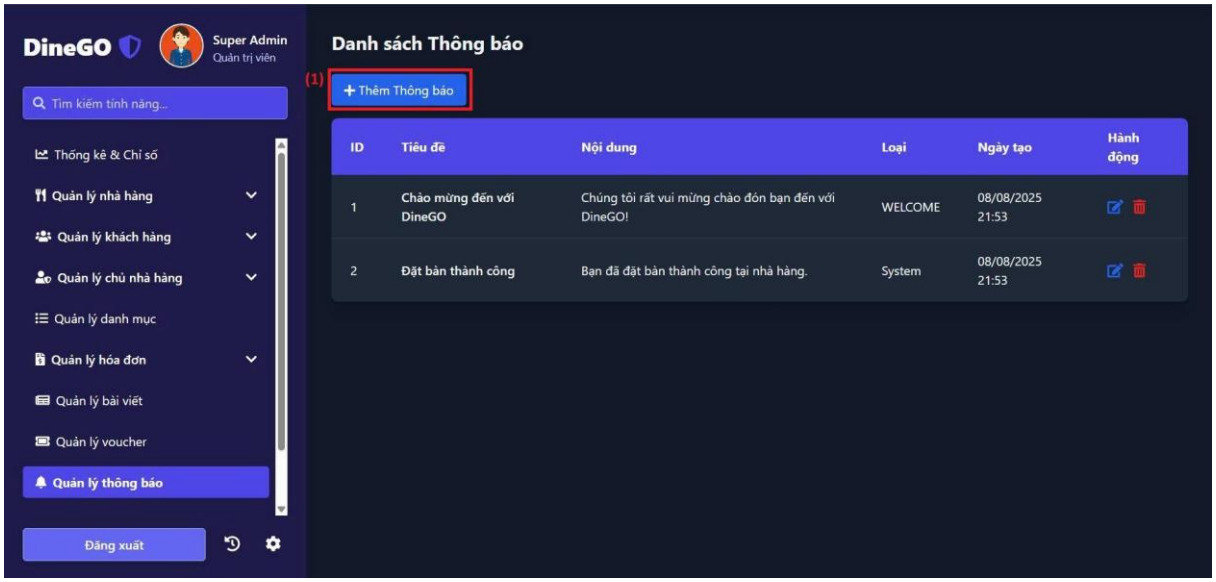


Figure 350: Notification page in DineGO

This feature allows the Admin to create a new system notification.

- (1) [+ Thêm thông báo] button to open the add notification form.

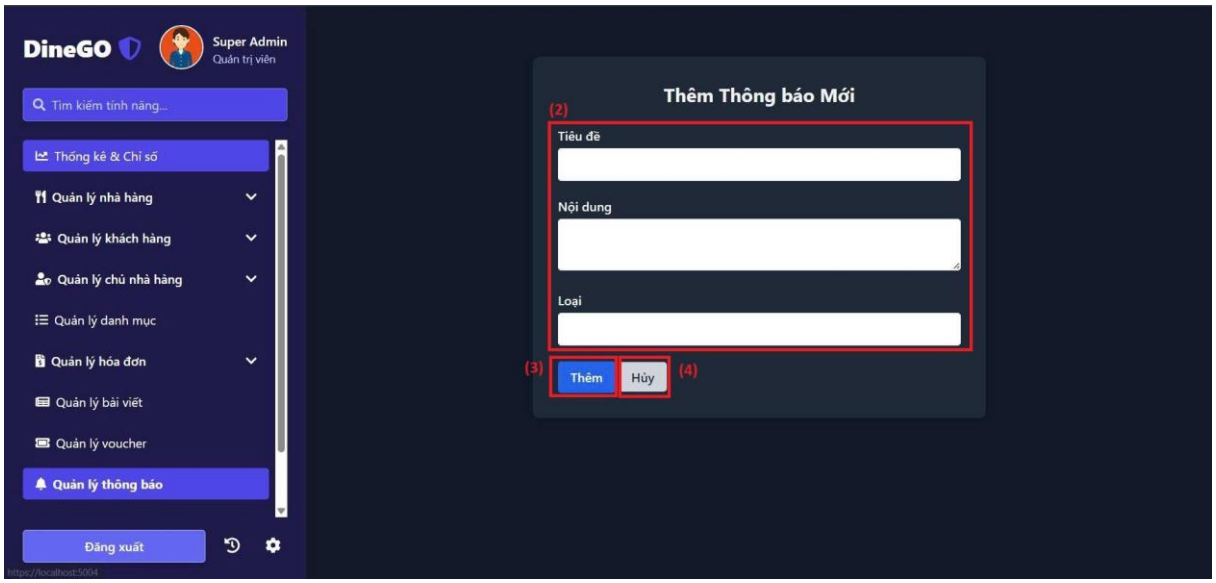


Figure 351: Add notification form of DineGO

- (2) Form fields to enter notification information
- (3) [Thêm] button to save the new notification.
- (4) [Hủy] button to discard changes and return to the notification list.
    - All fields must be filled in before adding.
    - If creation is successful, the new notification is added to the list.
    - If creation fails, an error message should be displayed.

iii. Update notification

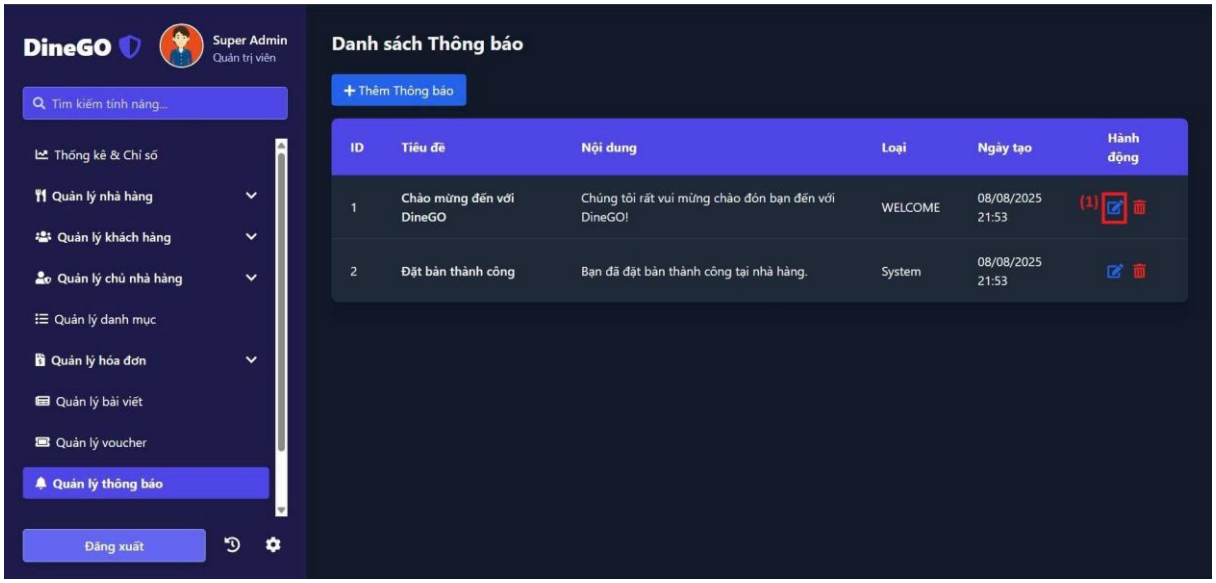


Figure 352: Notification page in DineGO

This feature allows the Admin to edit an existing notification

- (1) [Edit icon] next to the notification you want to modify.

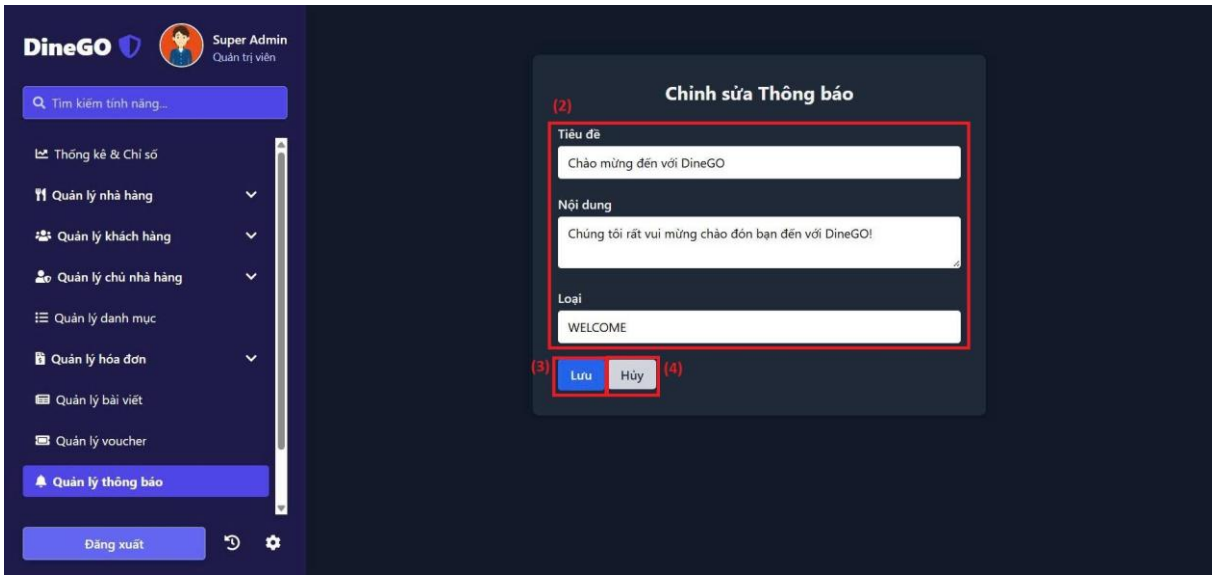


Figure 353: Update notification form of DineGO

- (2) Edit form fields want to edit
- (3) [Lưu] button to confirm and update the notification.
- (4) [Hủy] button to abort changes.
    - Title field is required and must not exceed the character limit.
    - "Nội dung" is required.
    - "Loại" must be a valid notification type supported by the system.
    - If the update is successful, the modified notification is saved and reflected in the list.
    - If the update fails, an error message should be displayed.

iv. Delete notification

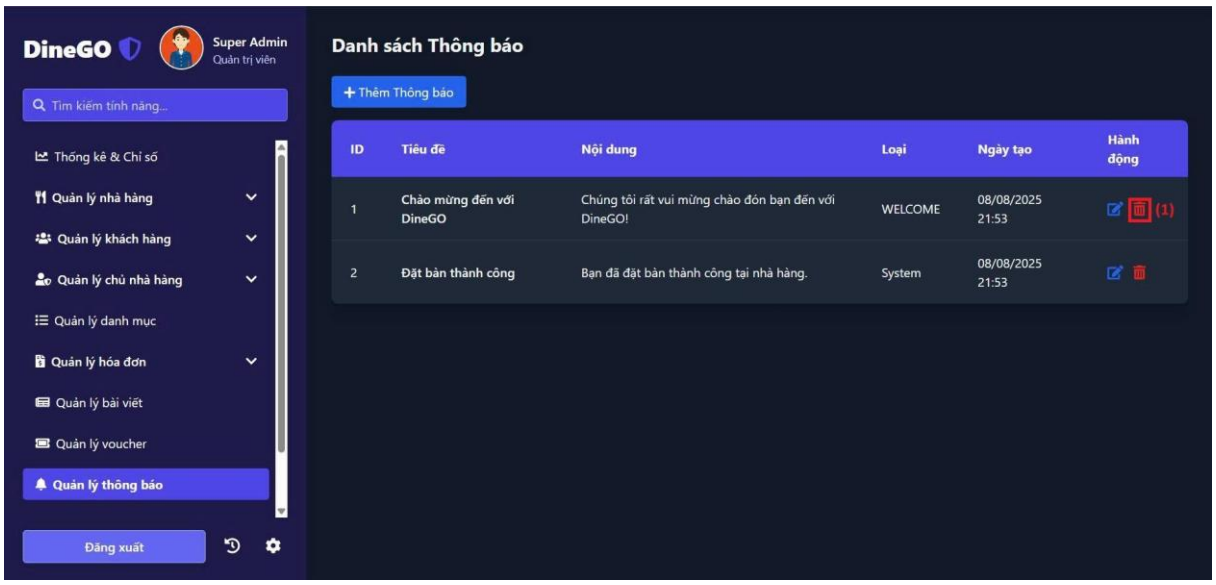


Figure 354: Notification page in DineGO

This feature allows the Admin to remove an existing notification.

- (1) [Delete icon] next to the notification you want to remove.

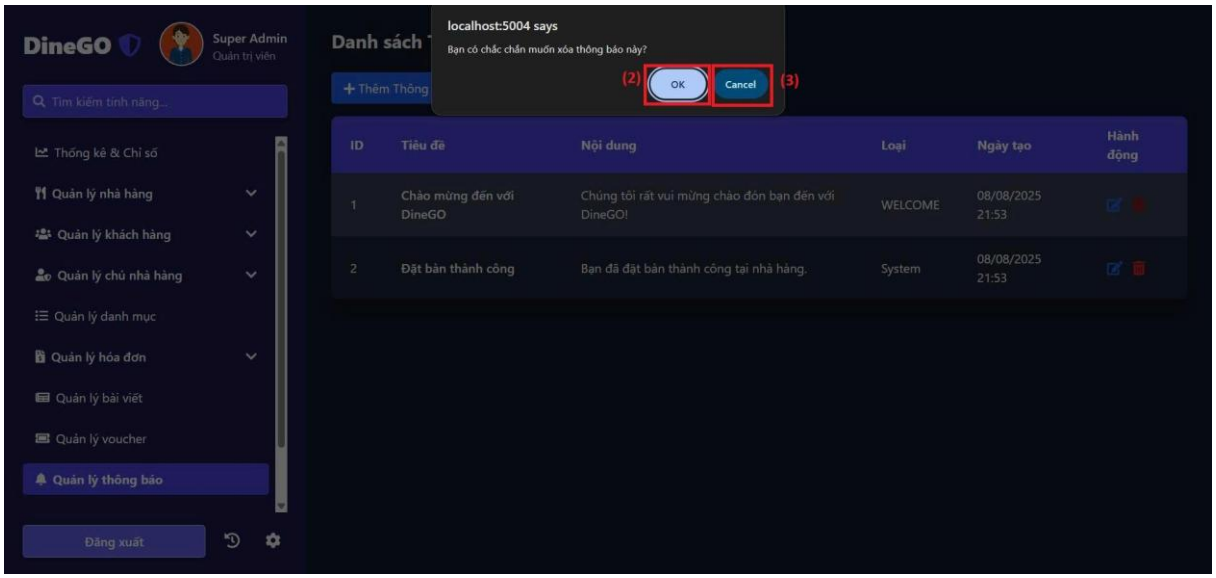


Figure 355: Delete notification in DineGO

- (2) [OK] button to confirm deletion.
- (3) [Cancel] button to abort deletion.
    - If deletion is successful, the notification is removed from the list.
    - If deletion fails, an error message should be displayed.

I. View dashboard reports and statistics

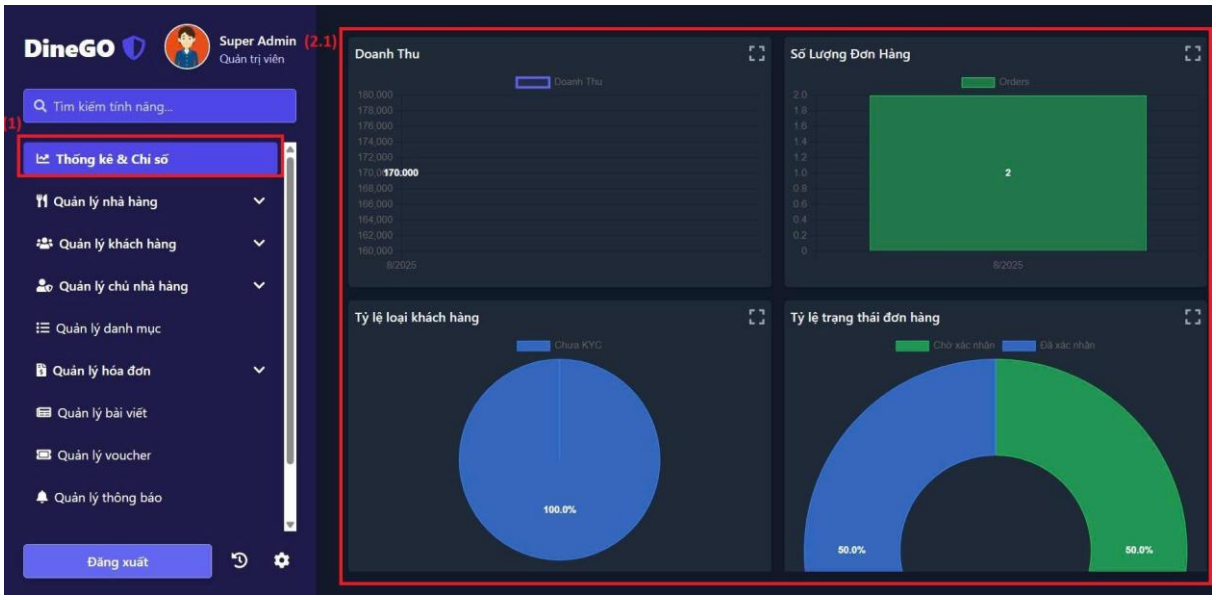


Figure 356: Dashboard and report in DineGO

This feature allows the Admin to view system statistics and KPIs.

- (1) [Thống kê & Chỉ số] menu item to open the dashboard.
- (2.1) Charts showing revenue, order quantity, customer type ratio, and order status ratio.
- (2.2) Charts showing top 5 restaurants by revenue, revenue comparison by service, and customer group analysis.
    - Dashboard provides real-time business insights for decision-making.

m. View activity logs

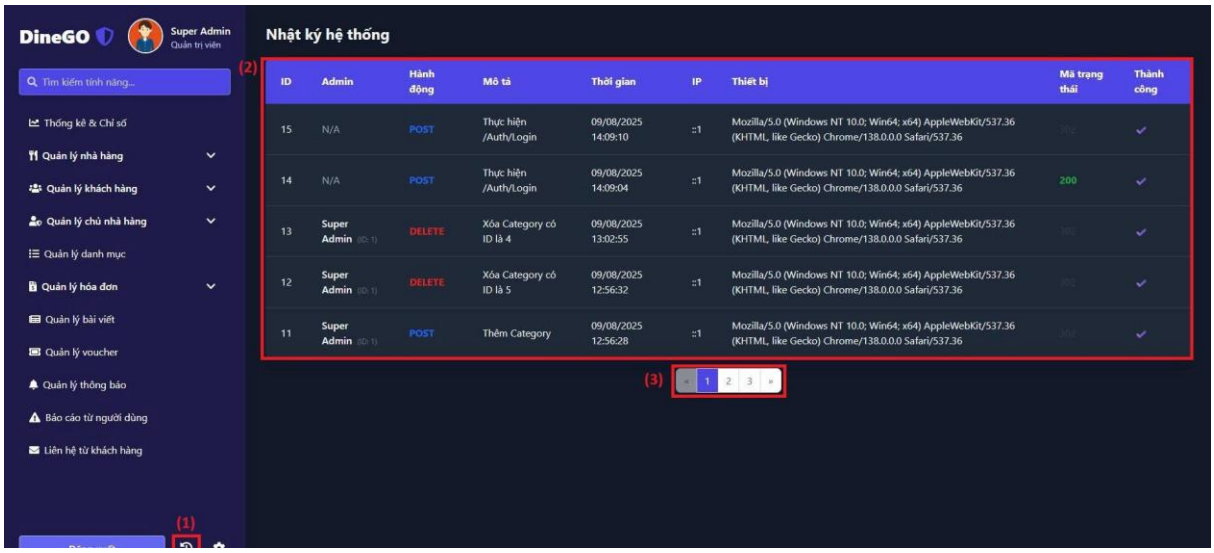


Figure 357: View activity log in DineGO

This feature allows the Admin to view the system's activity log.

- (1) [Activity log icon] in the sidebar to access the log page.
- (2) View the list of actions performed in the system
- (3) Use pagination controls to navigate through log entries.
    - The log list should display the most recent actions first

3.3.2 Customer feature

a. Search restaurants

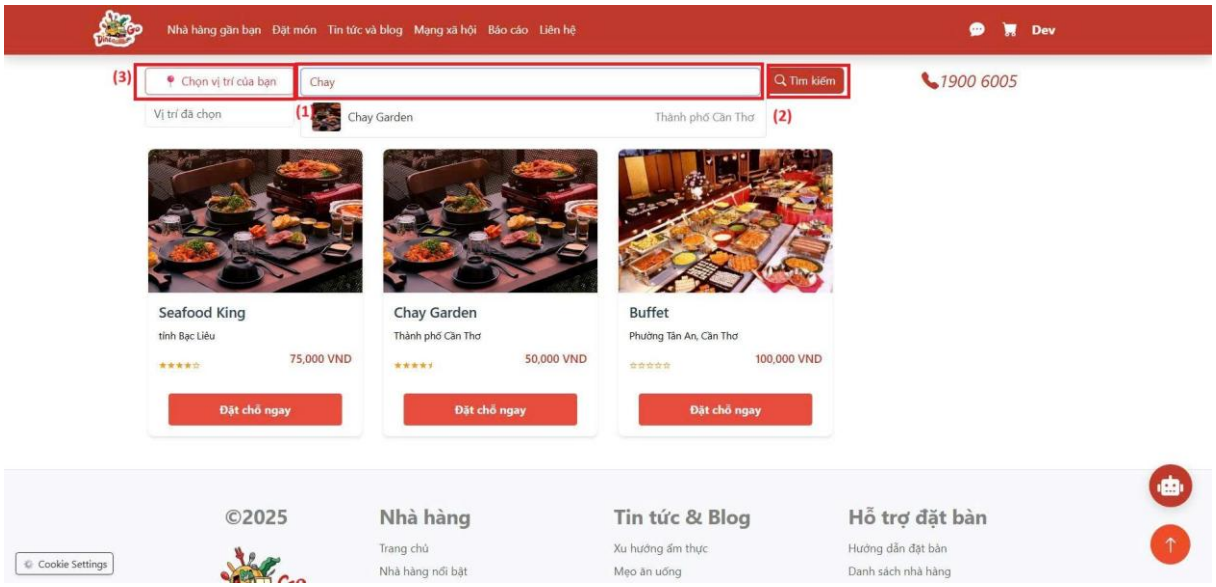


Figure 358: Search restaurant feature

This feature allows users to find restaurants based on their preferred keywords and location

- (1) Enter search keyword
- (2) [Tìm kiếm] button for search
- (3) [Chọn vị trí của bạn] button to select location

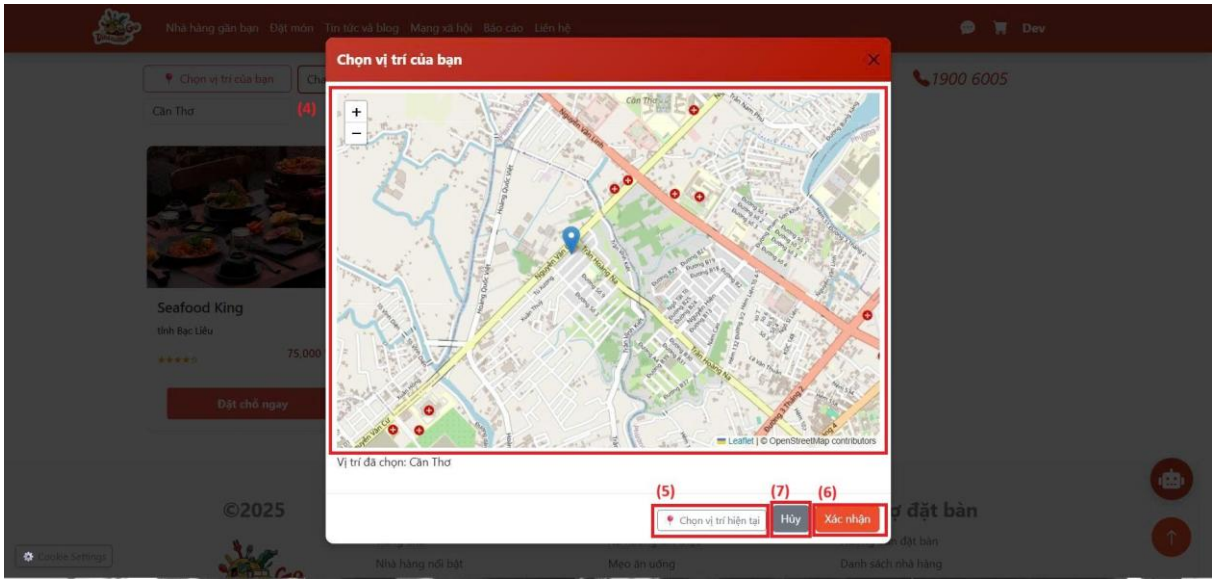


Figure 359: Select location feature

- (4) Select location on the map
- (5) [Vị trí hiện tại] button to set current position
- (6) [Xác nhận] button to confirm
- (7) [Hủy] button to close the location window without applying any changes

b. View restaurant detail

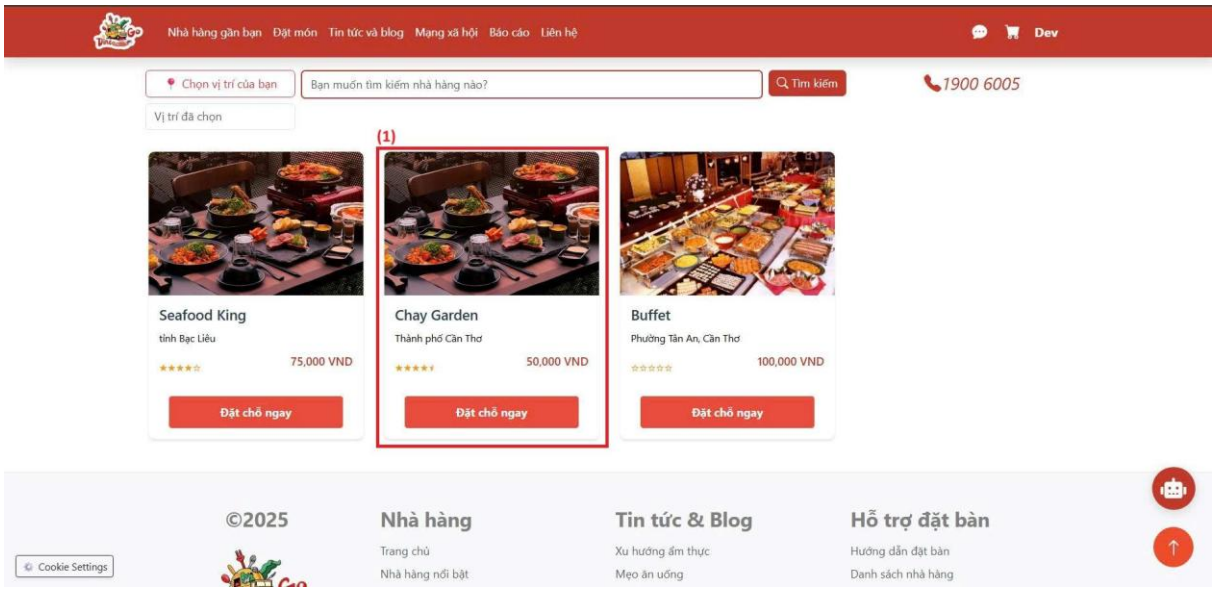


Figure 360: Restaurant page of DineGO

This page allows users to view detailed information about a specific restaurant

- (1) [restaurant card] to view its detail.

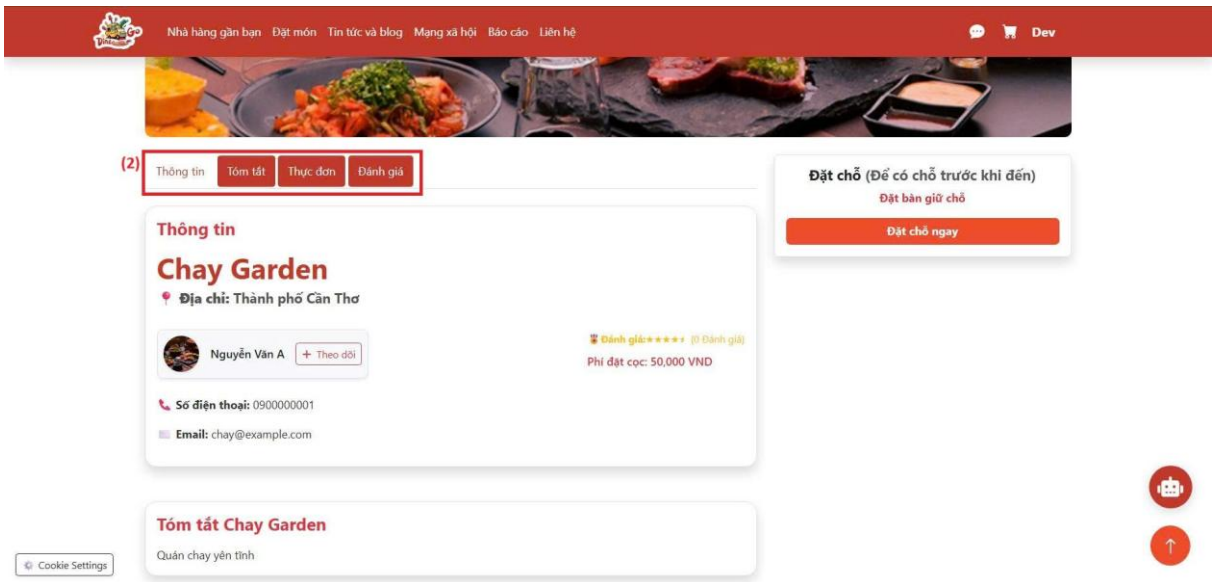


Figure 361: Restaurant detail page of DineGO

- (2) In the restaurant detail page, click the available tabs to view different information:
    - Tabs must be clickable and load the corresponding content without refreshing the whole page.

c. Browse food menus

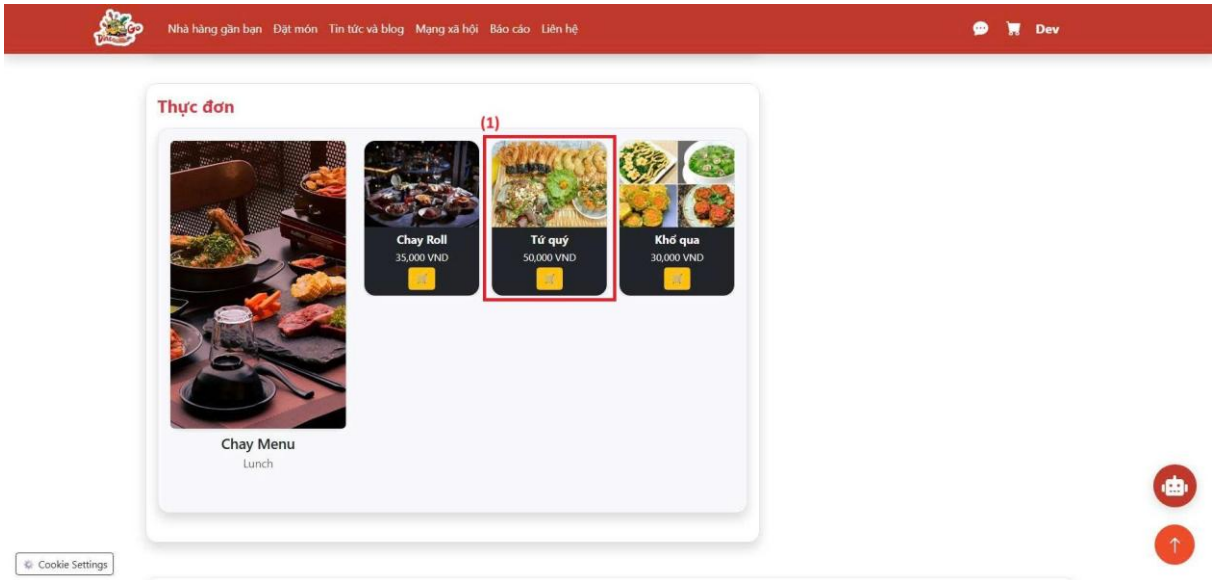


Figure 362: Menu of restaurant page of DineGO

This feature allows users to explore the restaurant's menu, view item detail

- (1) Select a food item from the menu list

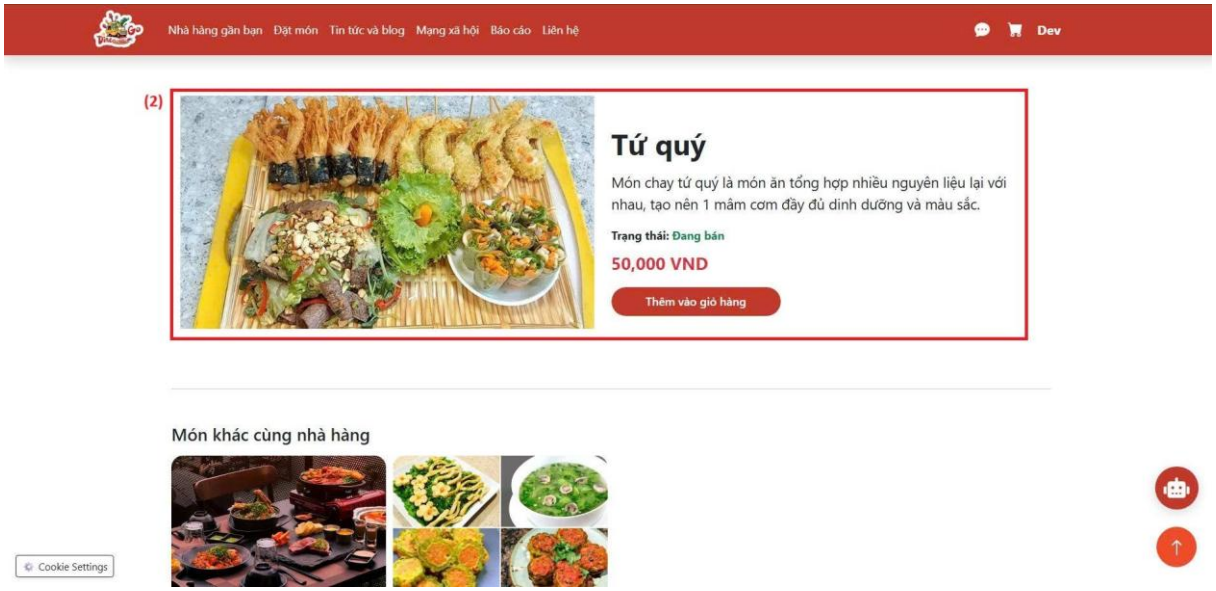


Figure 363: Food detail page of DineGO

- (2) View food detail

d. Mange shopping cart

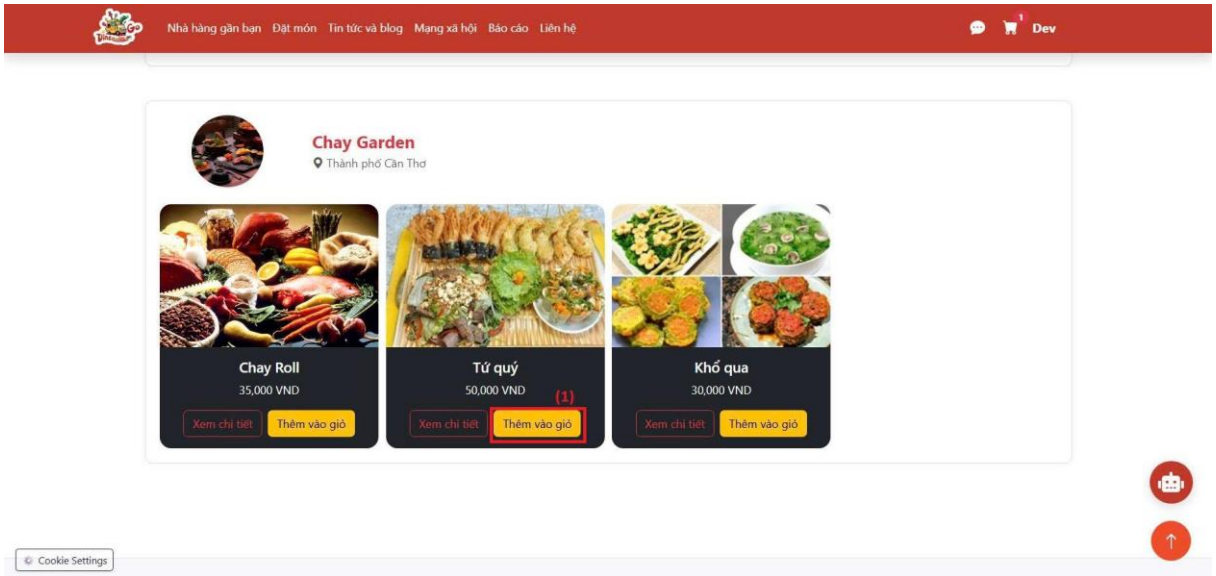


Figure 364: Add to cart button on restaurant menu page

This feature allows customers to select food items from a restaurant's menu and add them to their shopping cart for purchase

- (1) [Thêm vào giỏ] button to add an item to the cart

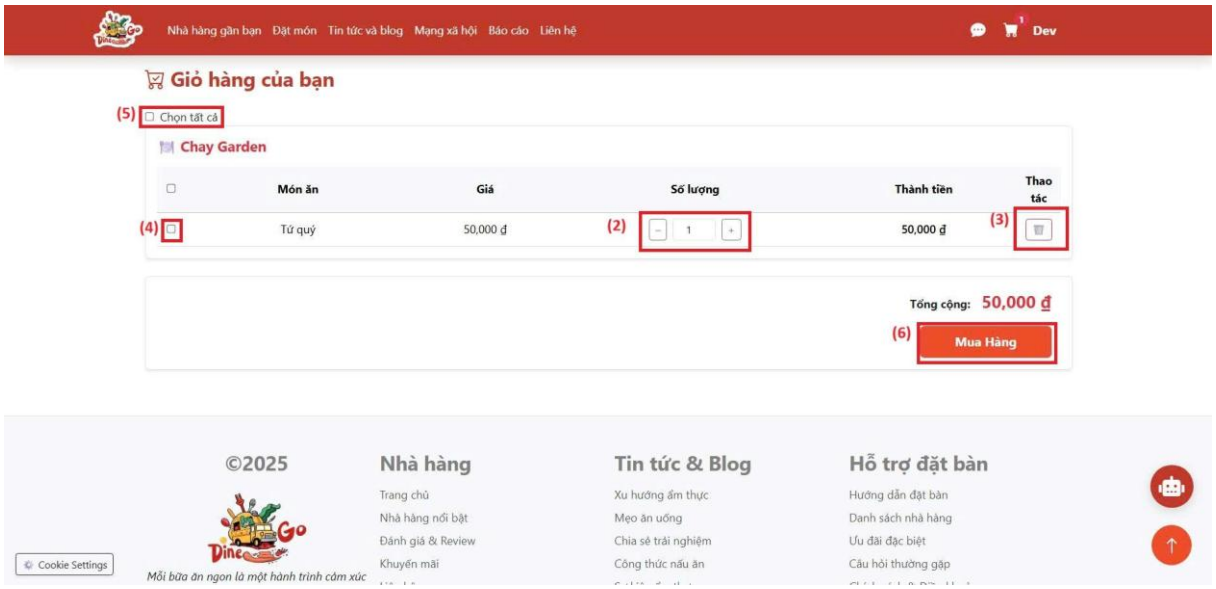


Figure 365: Shopping cart page of DineGO

- (2) (–) and (+) buttons adjust quantity in the cart
- (3) [Trash bin] icon to remove an item from the cart
- (4) Check the box next to a food item to mark it for purchase.
- (5) Select all items

- (6) [Mua hàng] button to purchase selected items
    - A user must be logged in to add items to the cart. If not, clicking "Thêm vào giỏ" redirects to the login page.
    - “Số lượng” must be at least 1 and cannot exceed available stock.
    - The cart must contain at least one selected item to proceed with purchase.
    - If an item is out of stock, it will be marked as unavailable and cannot be added to the cart.
    - The total price updates automatically based on quantity changes

e. Track order delivery

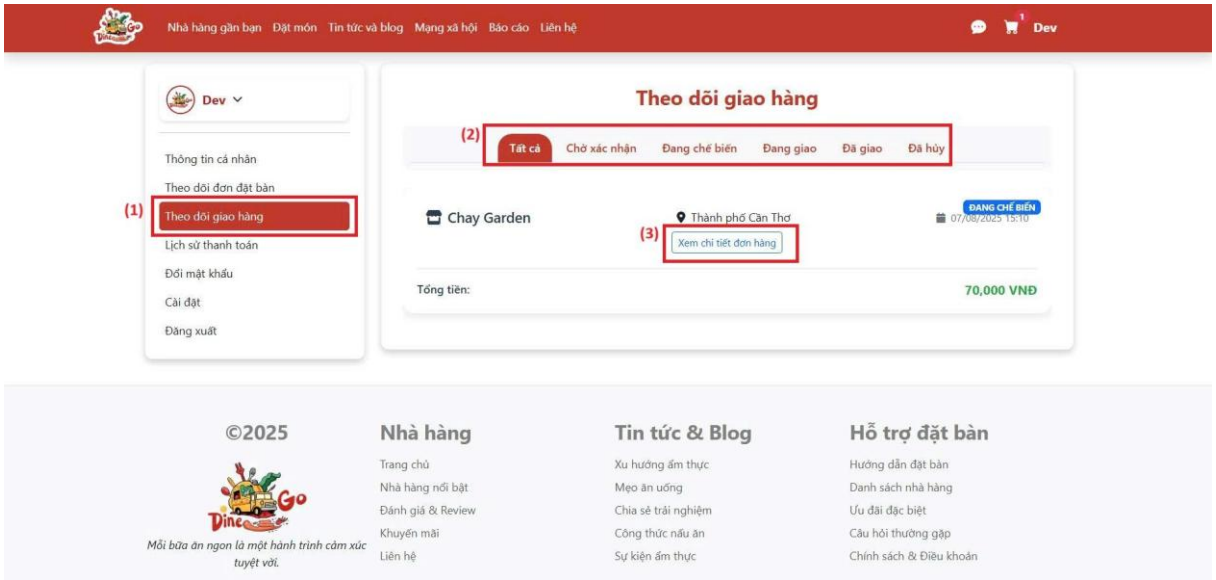


Figure 366: Track order delivery page of DineGO

This feature allows customers to monitor the delivery status of their food orders in real time

- (1) [Theo dõi giao hàng] tab from the account menu
- (2) Filter orders by status
- (3) View order detail
    - Only logged-in users can access the order tracking page.
    - The order list displayed is specific to the logged-in user.
    - The status filter must correspond to the actual order status in the system.

f. Make table reservations

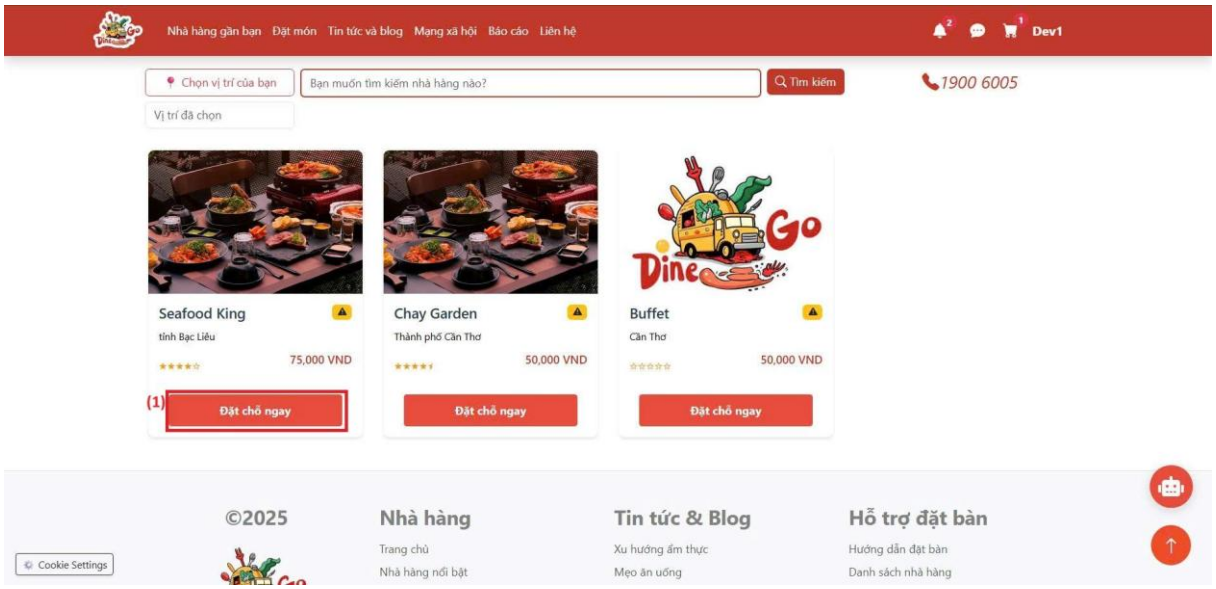


Figure 367: Make table reservation feature of DineGO

This feature allows the customer to reserve a table at a restaurant in advance.

- (1) [Đặt chỗ ngay] button on the restaurant card to open the restaurant detail page for booking.

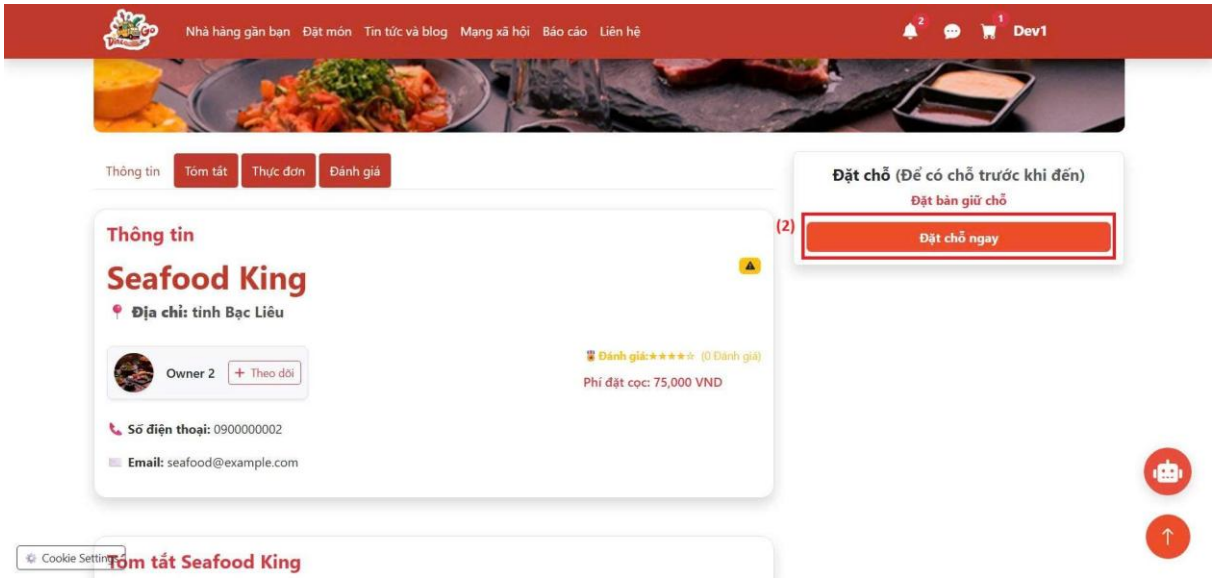


Figure 368: Restaurant detail page

- (2) [Đặt chỗ ngay] button on the restaurant detail page to proceed to the reservation form.

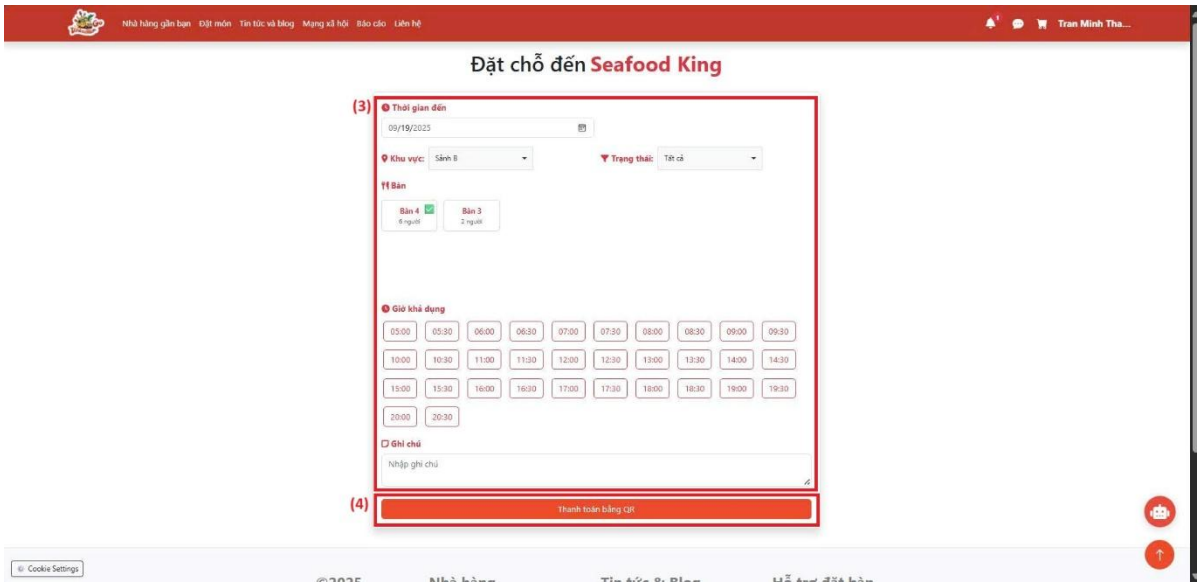


Figure 369: Make table reservation form of DineGO

- (3) Fill all fields
- (4) [Thanh toán bằng QR] button to proceed with payment and confirm the reservation.

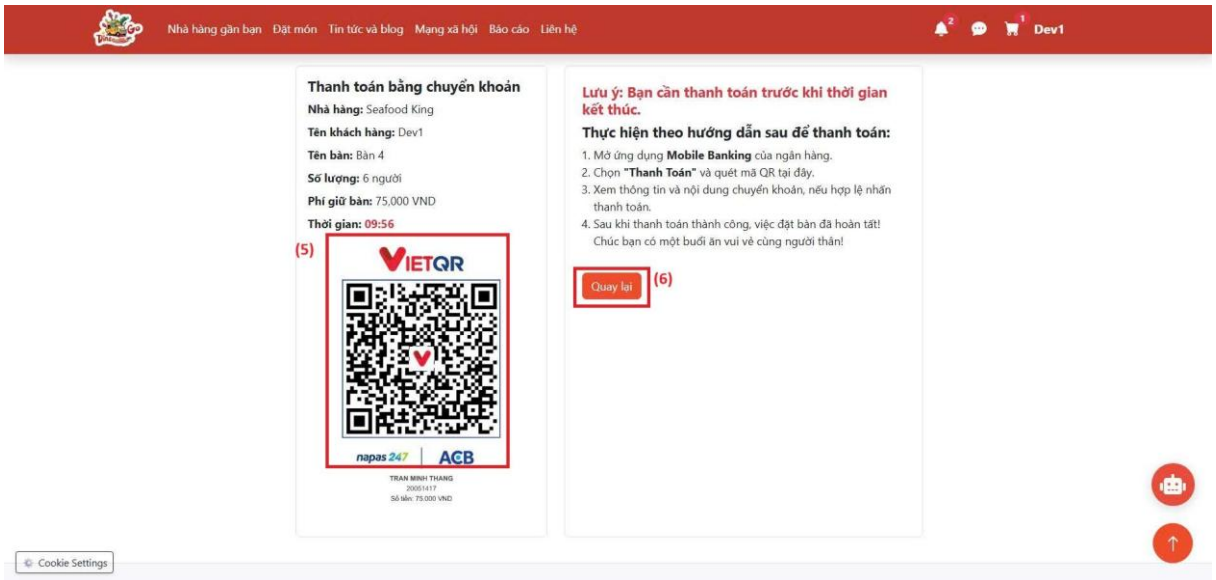


Figure 370: Payment for make table reservation of DineGO

- (5) QR Code for online payment – Scan using a mobile banking app to pay the deposit.
- (6) [Quay lại] button to return to the previous page without completing the payment.
    - Thời gian đến is required and must be a valid future time.
    - Khu vực selection is mandatory.
    - Bàn must be selected and must be in "Đang trống" status.
    - Ghi chú is optional but limited to 200 characters.
    - Payment must be completed before the countdown timer ends.
    - If the reservation is successful, the table status updates to "Đang chờ thanh toán" or "Đã đặt" depending on the payment state.
    - If the process fails, an error message is displayed, and the reservation is not saved

g. View reservation history

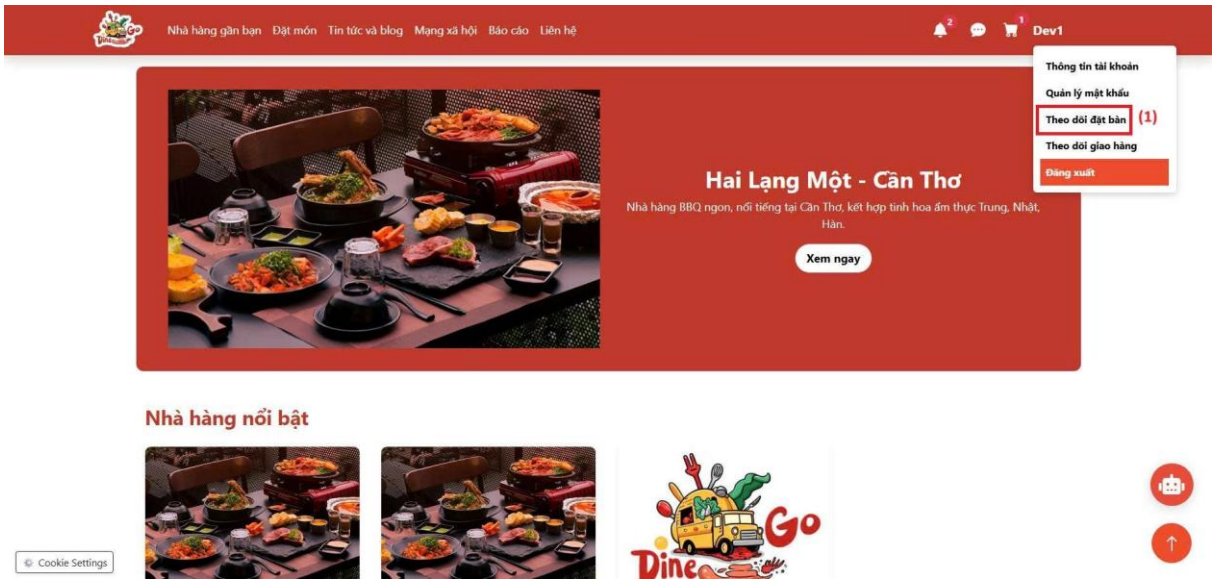


Figure 371: Profile page of DineGO

This feature allows the customer to view the history of their table reservations.

- (1) [Theo dõi đặt bàn] option in the user menu to navigate to the reservation history page.

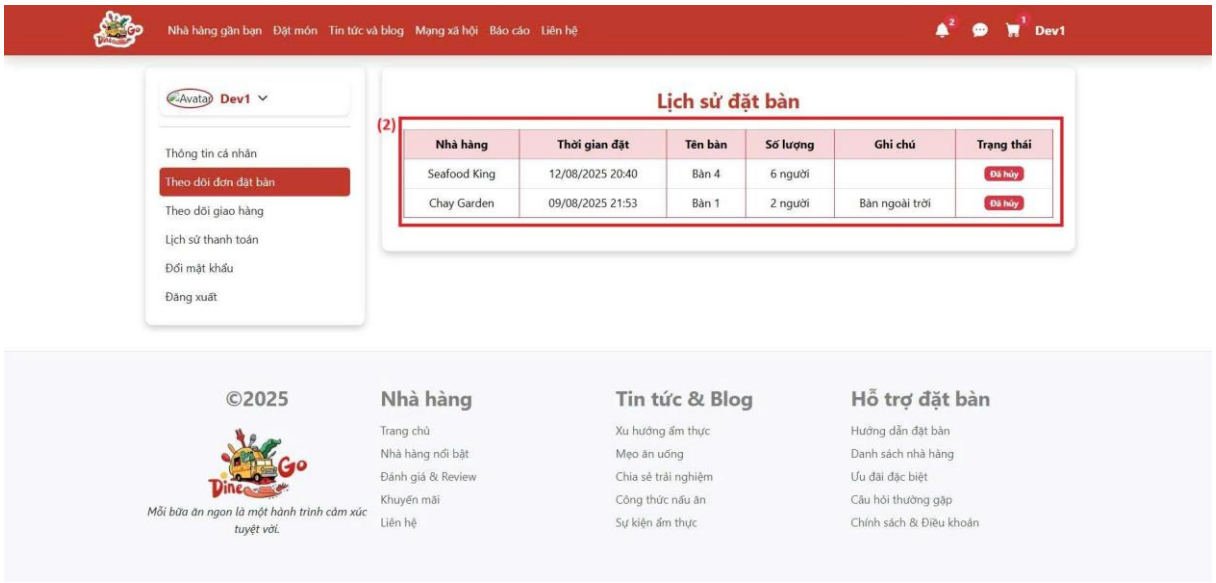


Figure 372: Reservation history of DineGO

- (2) "Lịch sử đặt bàn" table displaying detail for each reservation
  - The reservation list displays only reservations made by the currently logged-in customer.
  - Data must be sorted by reservation date in descending order (latest first).
  - Reservation status must accurately reflect the current booking status in the system.

h. Social features

i. Post

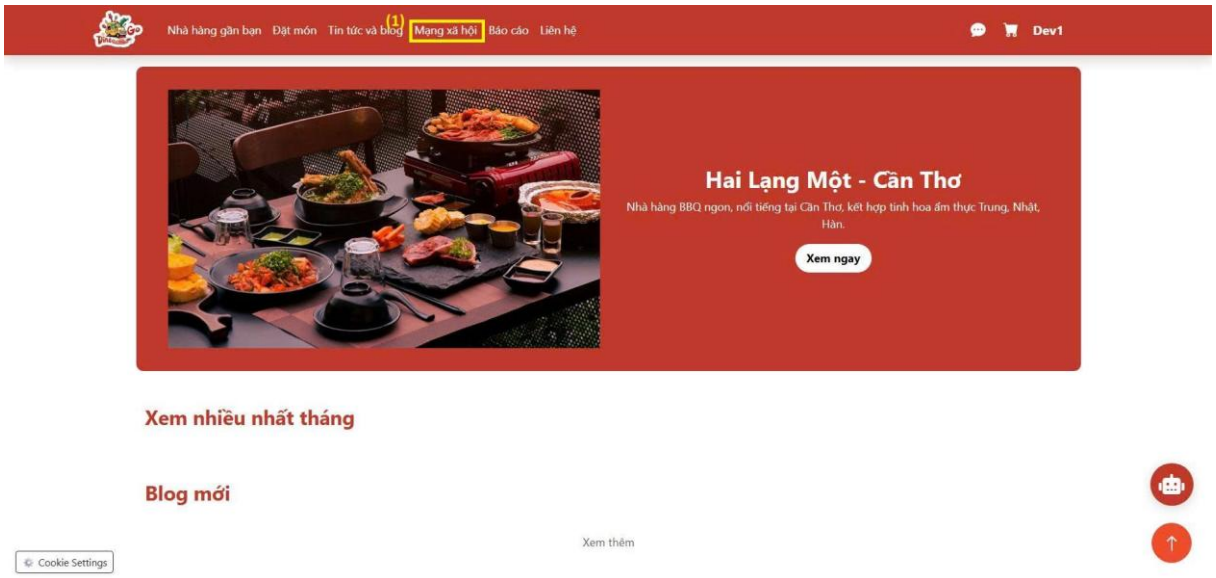


Figure 373: Homepage of DineGO

This feature allows users to write and share personal posts

- (1) [Mạng xã hội] tab to access the feed

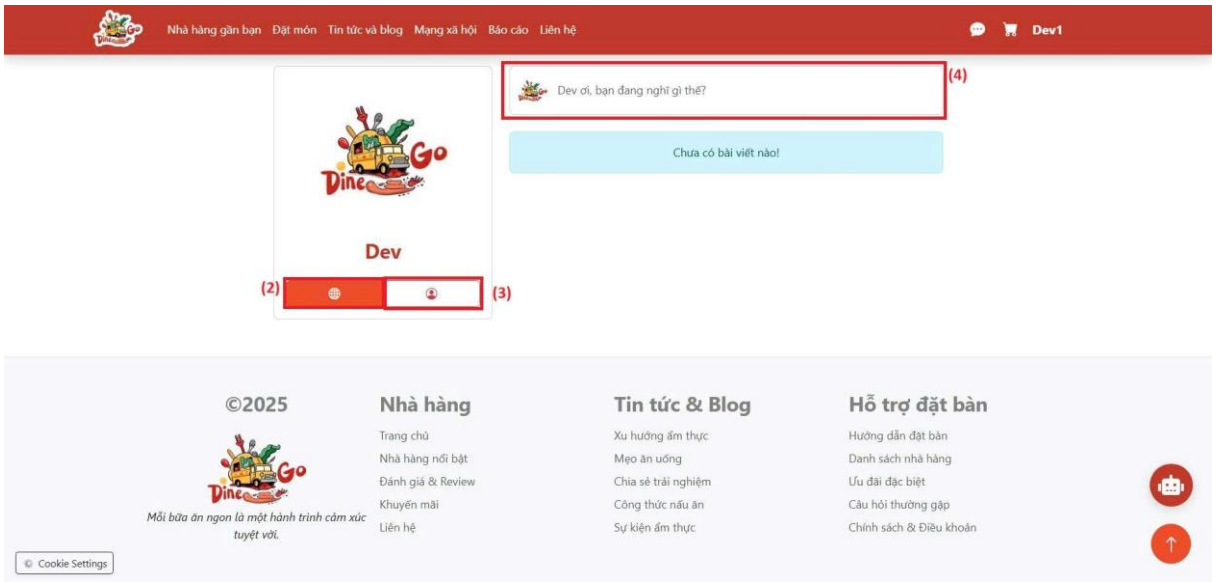


Figure 374: Post function of DineGO

- (2) Icon for all feed
- (3) Icon for only your feed
- (4) [Dev ơi, bạn đang nghĩ gì thế?] placeholder text to open the post creation modal

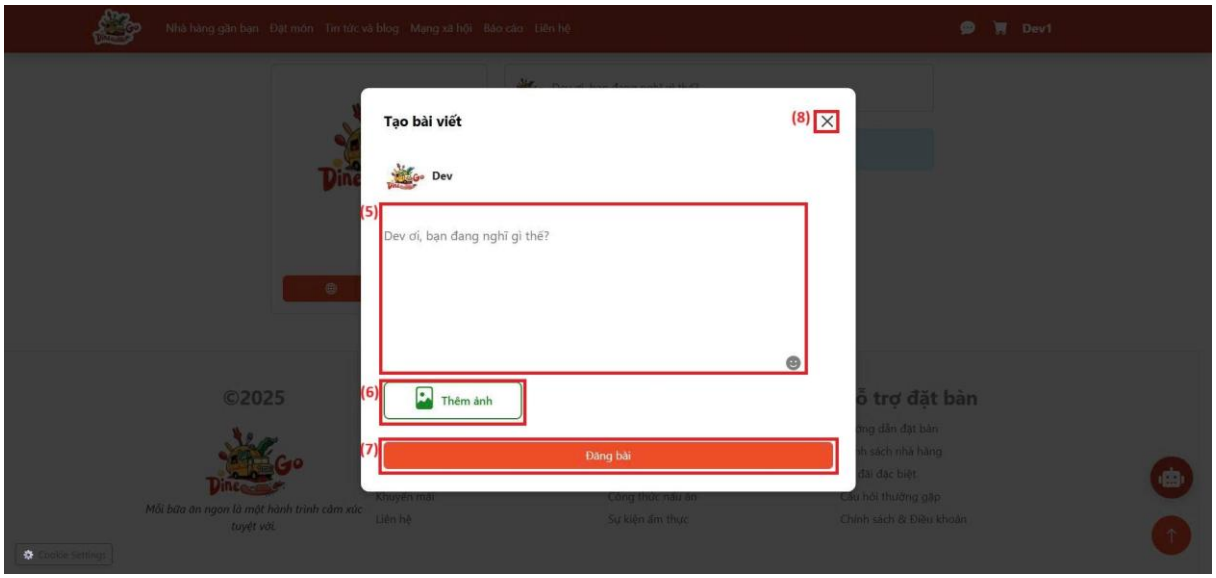


Figure 375: Form create post of DineGO

- (5) Enter post content
- (6) Add images
- (7) Submit the post
- (8)[X] button to close the modal
    - The post content cannot be empty.
    - The post content cannot be empty.
    - Only image files in accepted formats (e.g., JPG, PNG) can be uploaded.
    - After posting, the new post will appear on the user's timeline/feed.

ii. Comment

This feature allows users to leave comments on posts

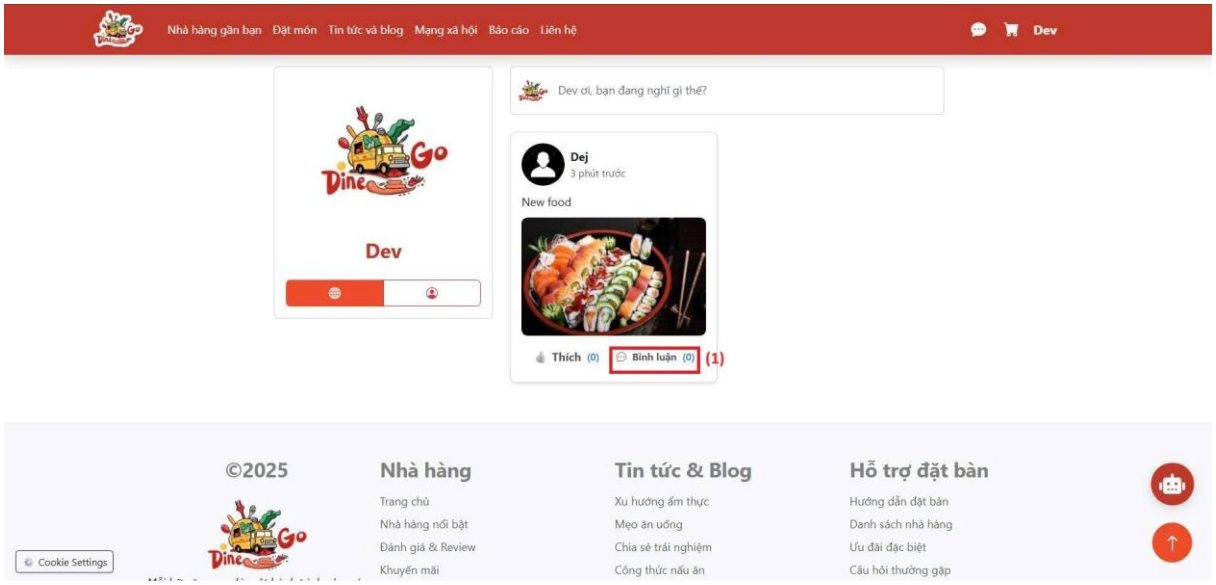


Figure 376: Social media page of DineGO

- (1) [Bình luận] button to comment

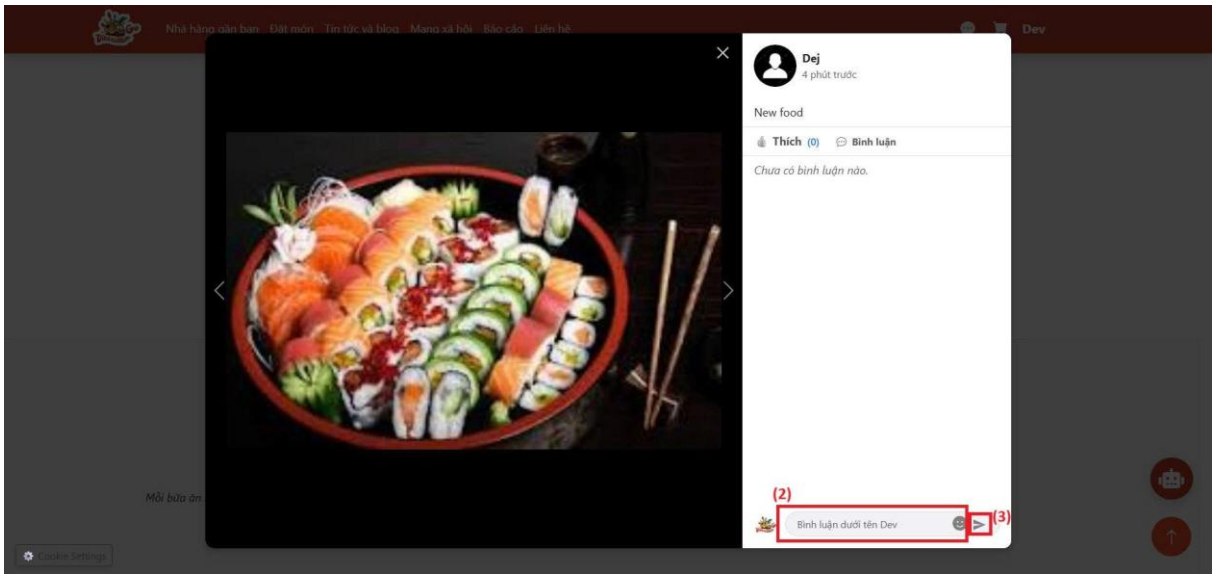


Figure 377: Feed in social media of DineGO

- (2) Enter comment
- (3) [Paper plane symbol] icon to submit the comment
    - Comments are posted under the logged-in user's name
    - Empty comments are not allowed; the system may show a warning if no content is entered.
    - Once submitted, the comment appears immediately under the post.

iii. Likes

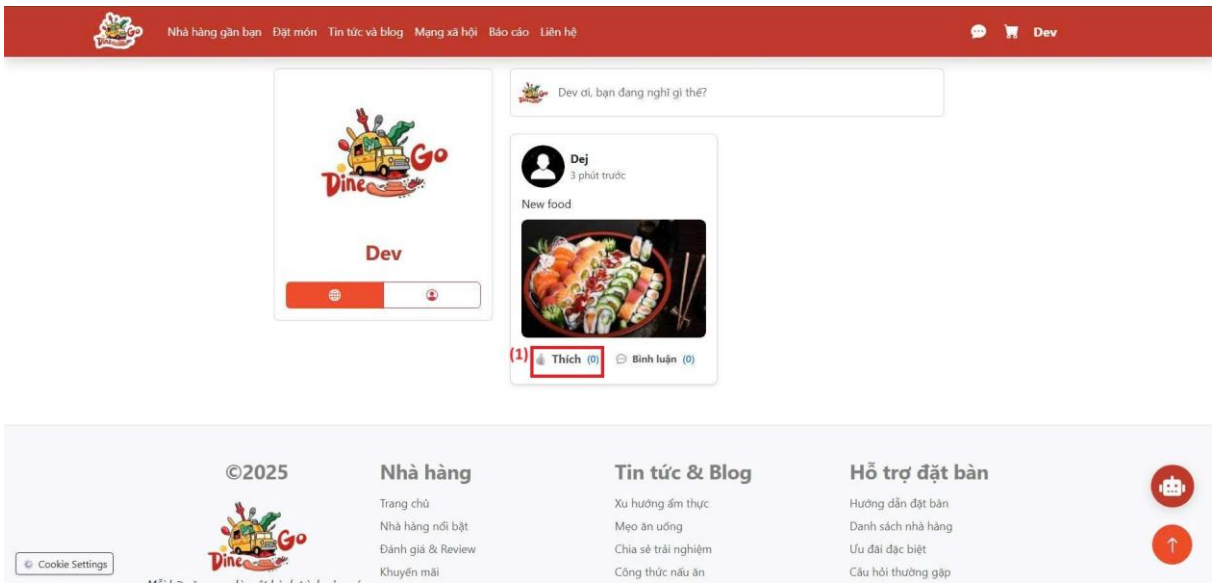


Figure 378: Social media page of DineGO

This feature allows users to like a post or choose from multiple reaction emojis

- (1) Hover [Thích] button to open the reaction options

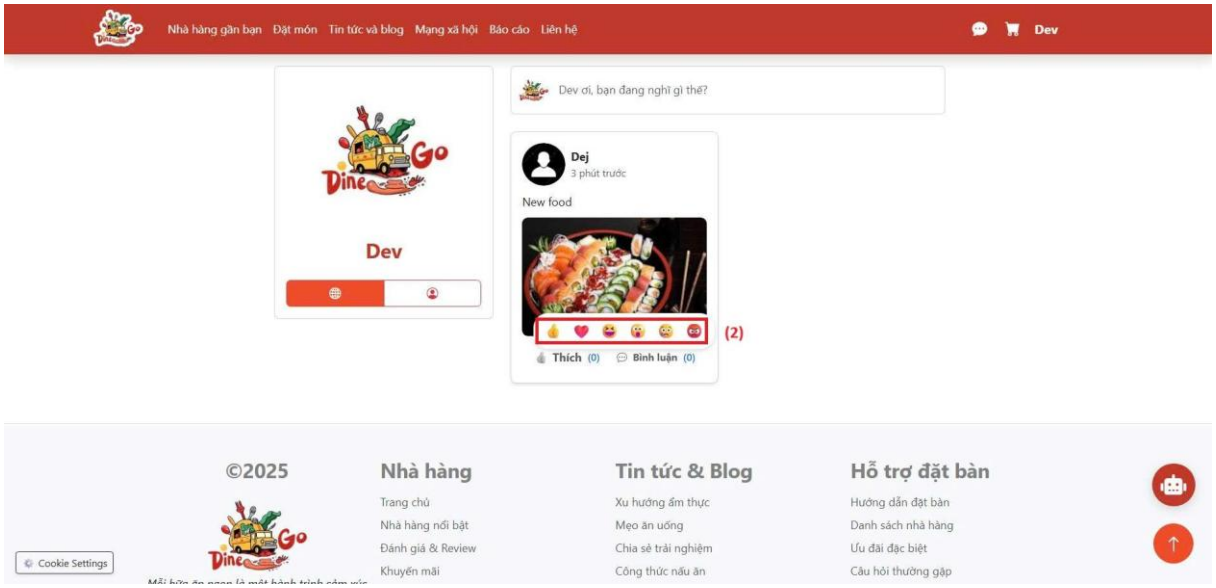


Figure 379: Social media page of DineGO

- (2) Choose a reaction
  - The chosen reaction will be recorded and displayed next to the post.
  - You can change your reaction by clicking the Like button again and selecting a different emoji.
  - Clicking the same reaction again may remove it
  - The reaction count updates in real-time.

iv. Chat with friends

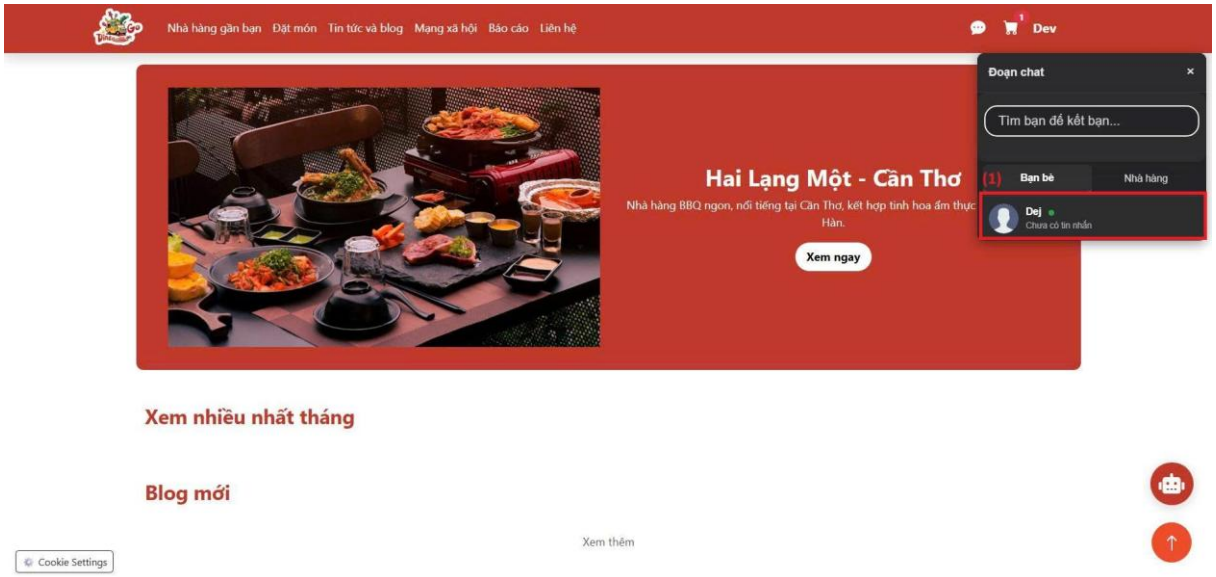


Figure 380: Friend list in chat panel

This feature allows users to send and receive messages with friends

- (1) Open the friend chat list

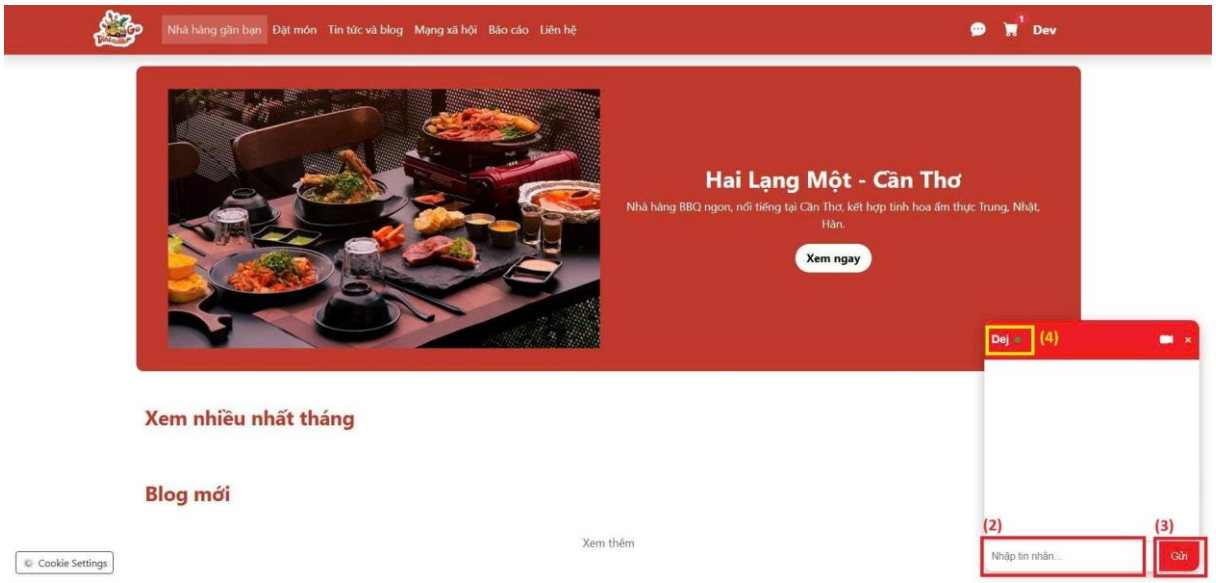


Figure 381: Sending message to a friend

- (2) Enter your message
- (3) [Gửi] button to send message
- (4) Check friend's online status
    - Message content cannot be empty.
    - Special characters and emojis are supported.
    - Internet connection is required for the message to be sent.

v. Add friends

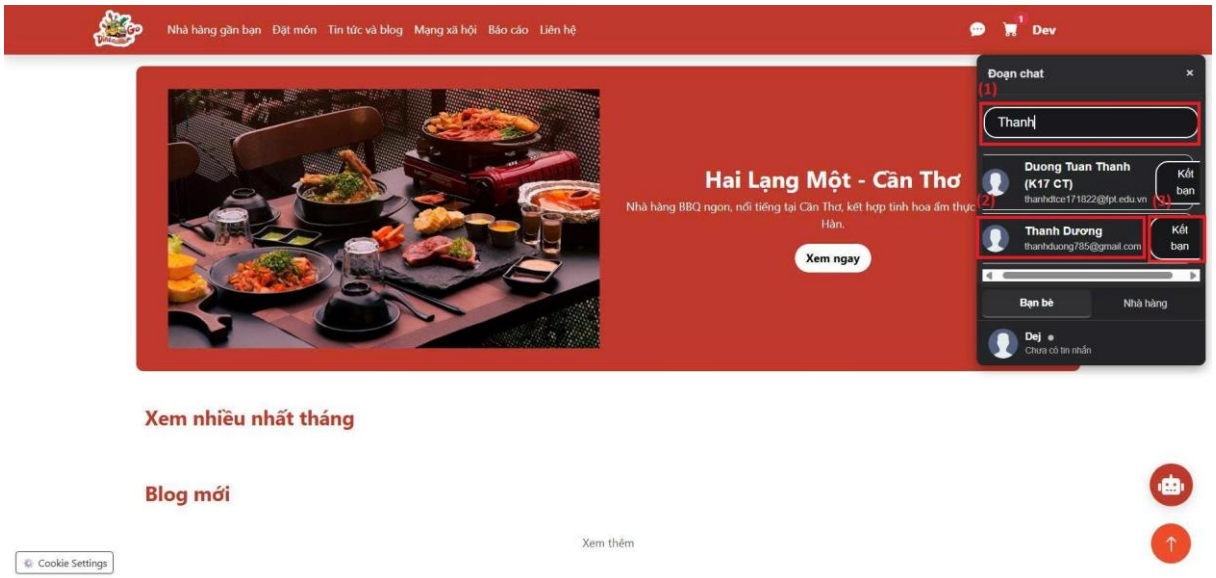


Figure 382: Add Friend screen

This feature allows users to search for other users and send them a friend request

- (1) Enter the friend's name in the search bar
- (2) View search results
- (3) [Kết bạn] button to add friend
    - You cannot send a friend request to yourself.
    - If a friend request is already pending or the user is already your friend, the Kết bạn button will be disabled.
    - Internet connection is required to send a friend request.

vi. Accept friends

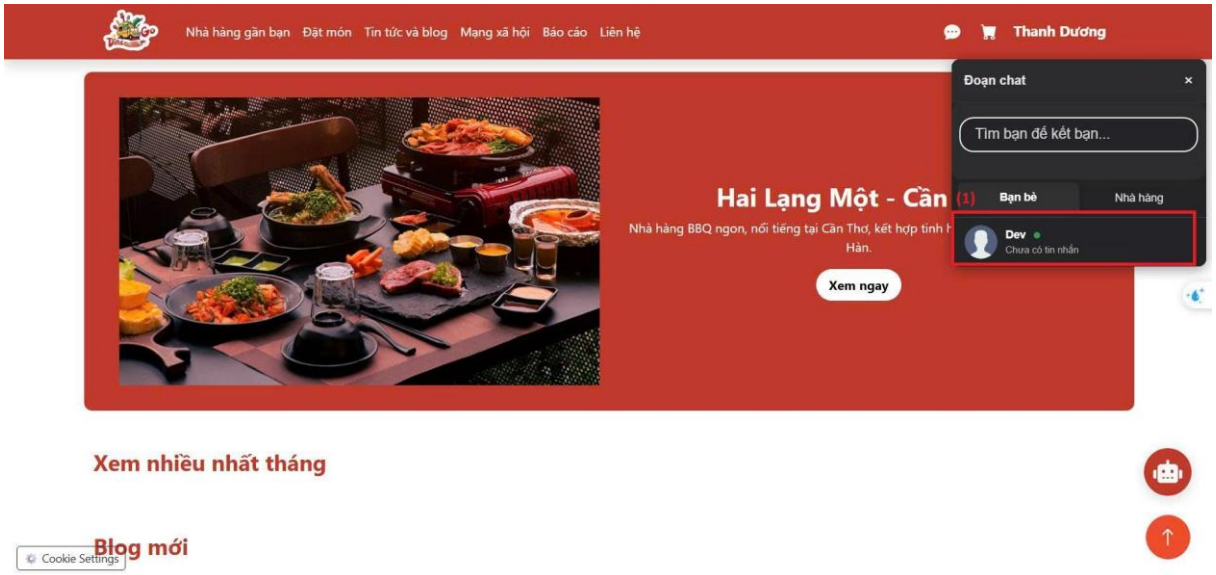


Figure 383: Friend list showing pending request

This feature allows users to accept a friend request and start chatting with the other person.

- (1) View pending friend request

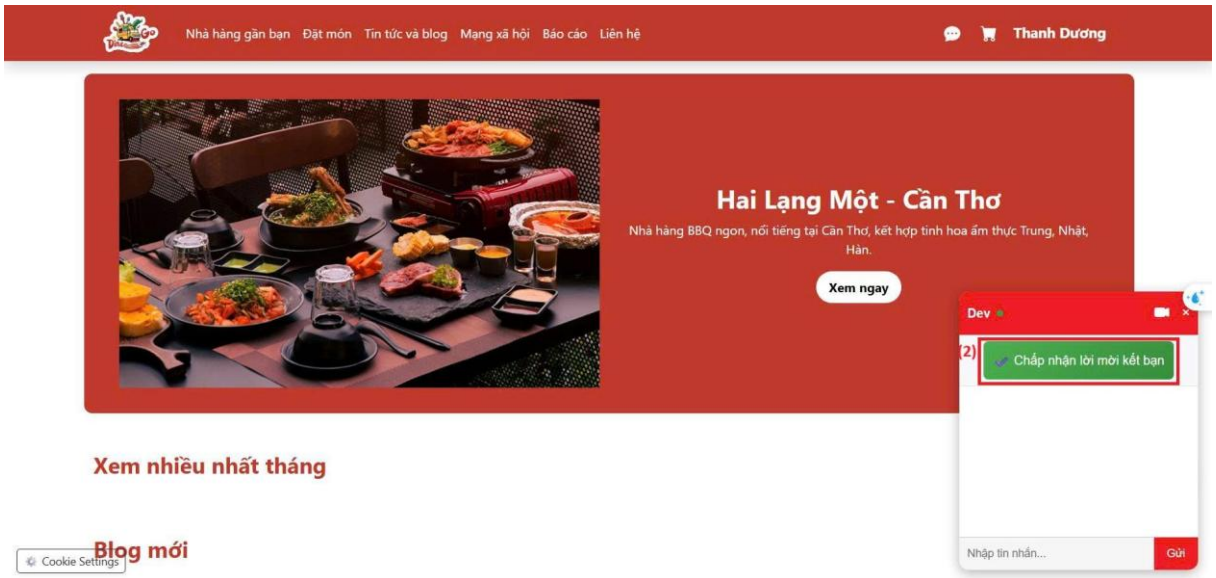


Figure 384: Chat window

- (2) [Chấp nhận lời mời kết bạn] button to accept the friend request
  - The user must have already sent a friend request before you can accept it.
  - After accepting, the system automatically establishes a two-way friendship so both users can chat freely.
  - The friend will now appear in your friends list, and messages can be exchanged instantly

i. Register as restaurant owner

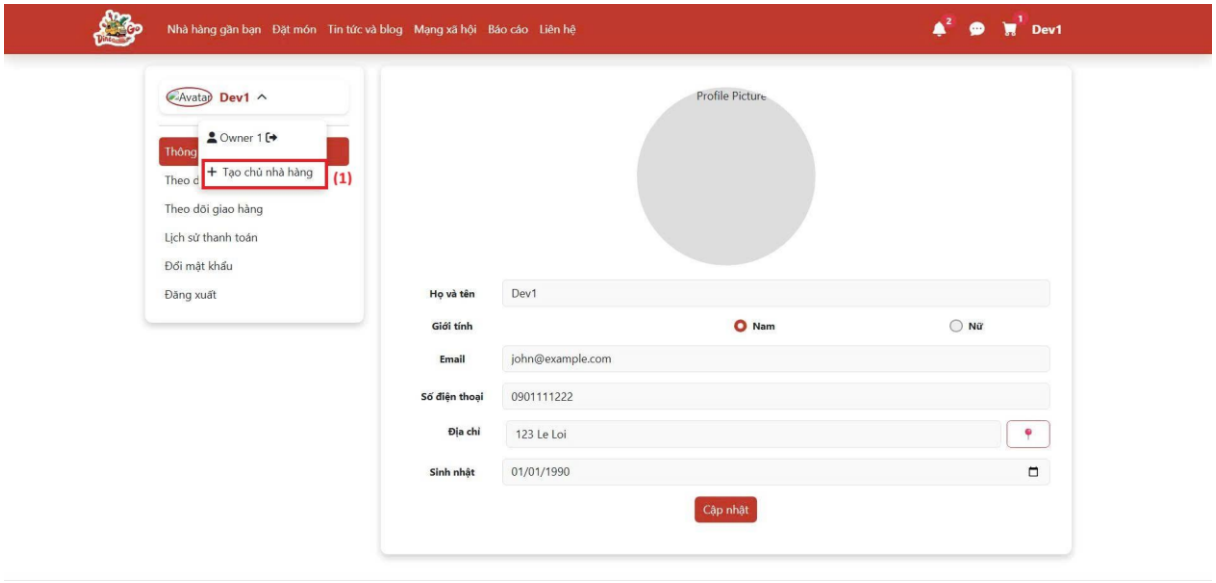


Figure 385: Profile page of DineGO

This feature allows a customer to register as a restaurant owner

- (1) [Tạo chủ nhà hàng] button in the profile menu to open the registration form.

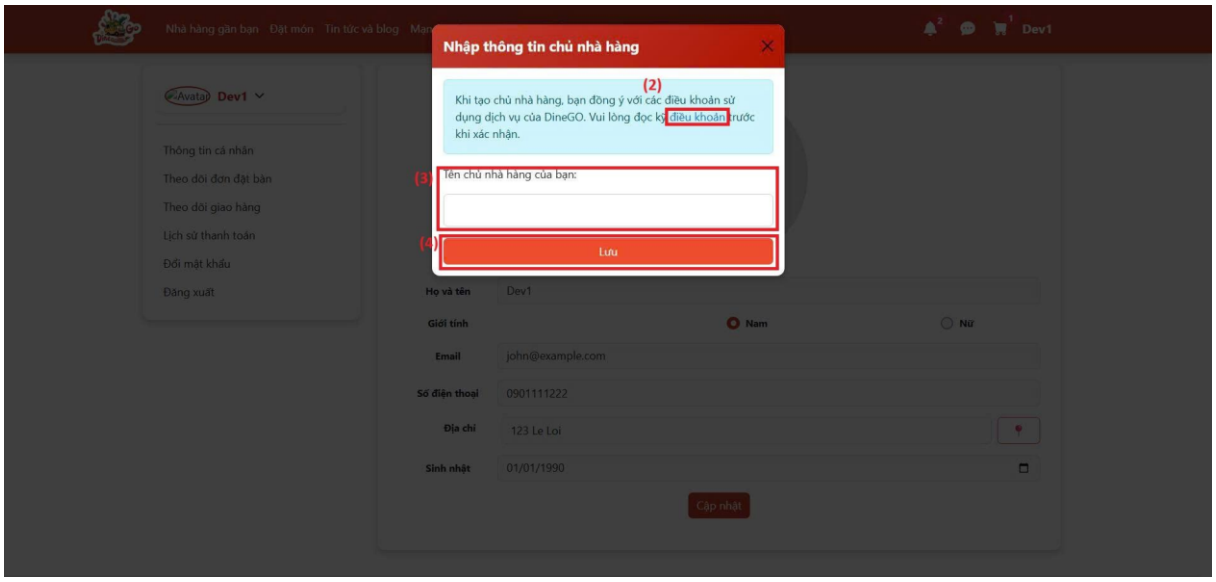


Figure 386: Register restaurant owner form of DineGO

- (2) Terms of Service notice with a link to view detailed terms before confirming registration.
- (3) Tên chủ nhà hàng – Enter the name of the restaurant owner (required).
- (4) [Lưu] button to submit the registration request.
    - The "Tên chủ nhà hàng của bạn" is required and must not be empty.
    - The "Tên chủ nhà hàng của bạn" must be unique in the system.
    - The user must agree to the terms of service before submitting.
    - If registration is successful, the system updates the user's account to include restaurant owner privileges.
    - If registration fails, an error message is displayed

j. Manage voucher

i. View voucher

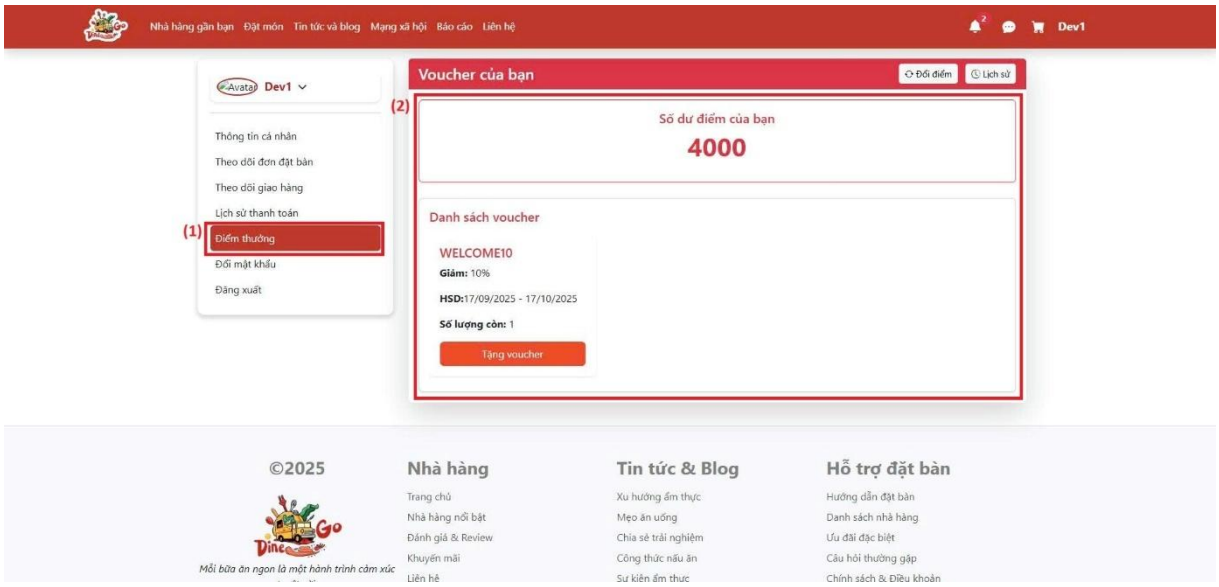


Figure 387: View Vouchers and Points

This feature allows the Customer to view their current point balance and available vouchers.

- (1) [Điểm thưởng] menu item in the account dropdown to open the Reward Points page.
- (2) Voucher panel:
    - "Số điểm của bạn" – Displays the customer's current points.
    - Voucher list – Shows all vouchers the customer owns, including detail such as discount rate, expiry date, and remaining quantity.
    - [Tặng voucher] button – Allows the customer to gift the voucher to another user

ii. Redeem points for vouchers

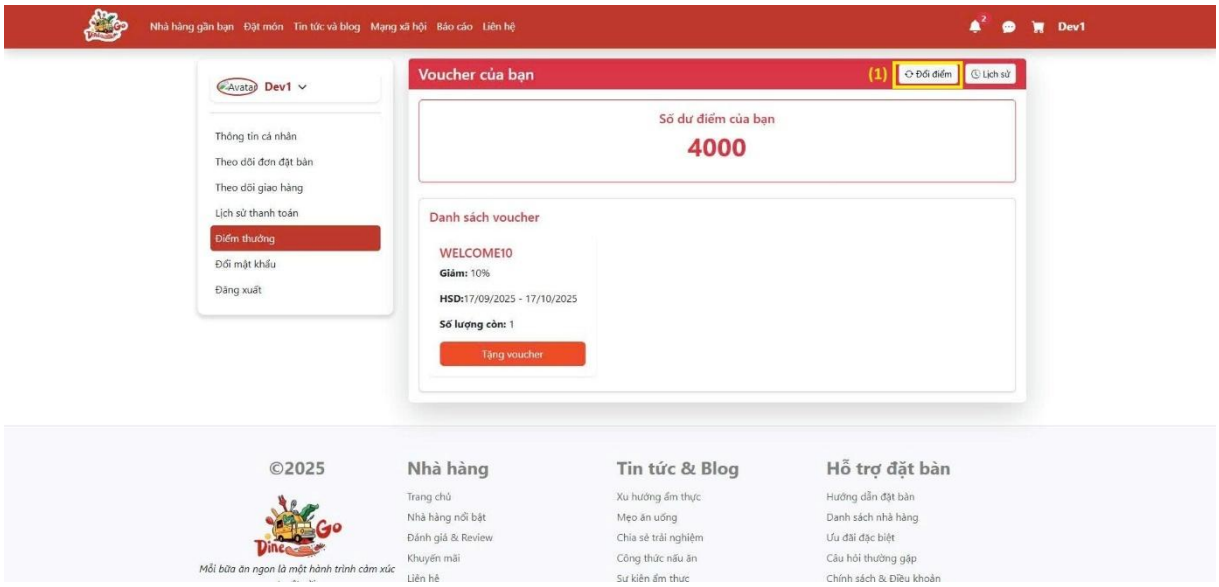


Figure 388 : Voucher page

This feature allows the Customer to exchange reward points for discount vouchers.

- (1) [Đổi điểm] button on the Voucher page to open the Redeem Points page.

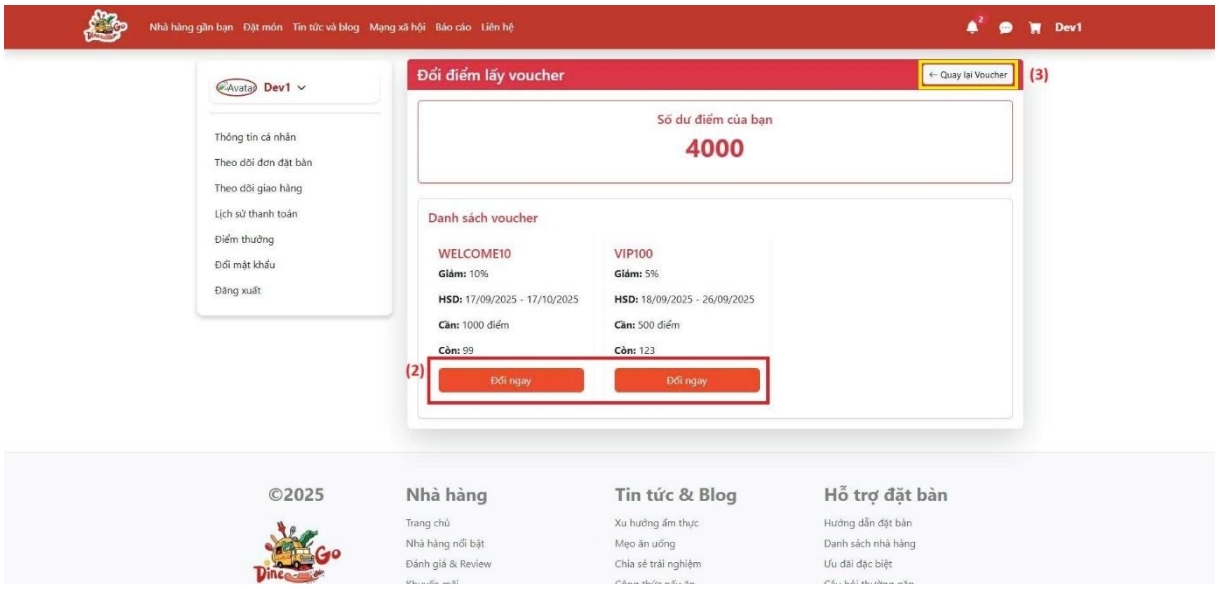


Figure 389: Reedeem point for vouchers

- (2) [Đổi ngay] button – Allows the Customer to redeem the selected voucher immediately.
- (3) [Quay lại Voucher] button – Returns to the main Voucher page.
    - The Customer must have enough points to redeem the voucher.
    - The voucher must still be available (quantity > 0).
    - Expired vouchers cannot be redeemed.
    - If redemption is successful, points will be deducted, and the voucher is added to the Customer's account.

iii. Point transaction history

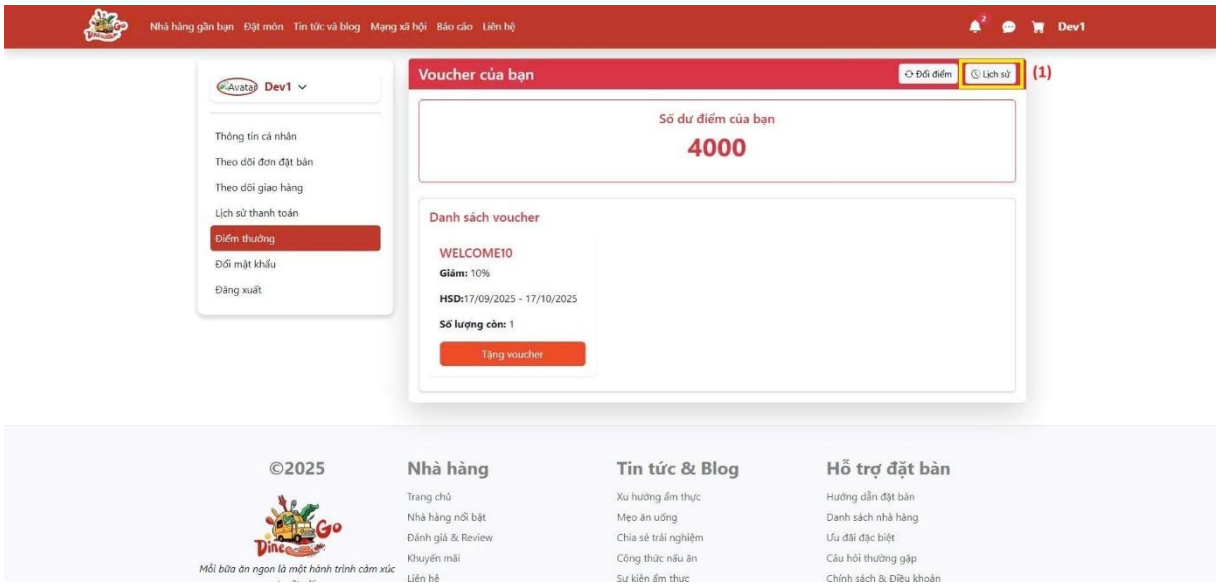


Figure 390: Voucher page

This feature allows the Customer to view their point transaction history

- (1) [Lịch sử] button on the Voucher page to open the point transaction history page.

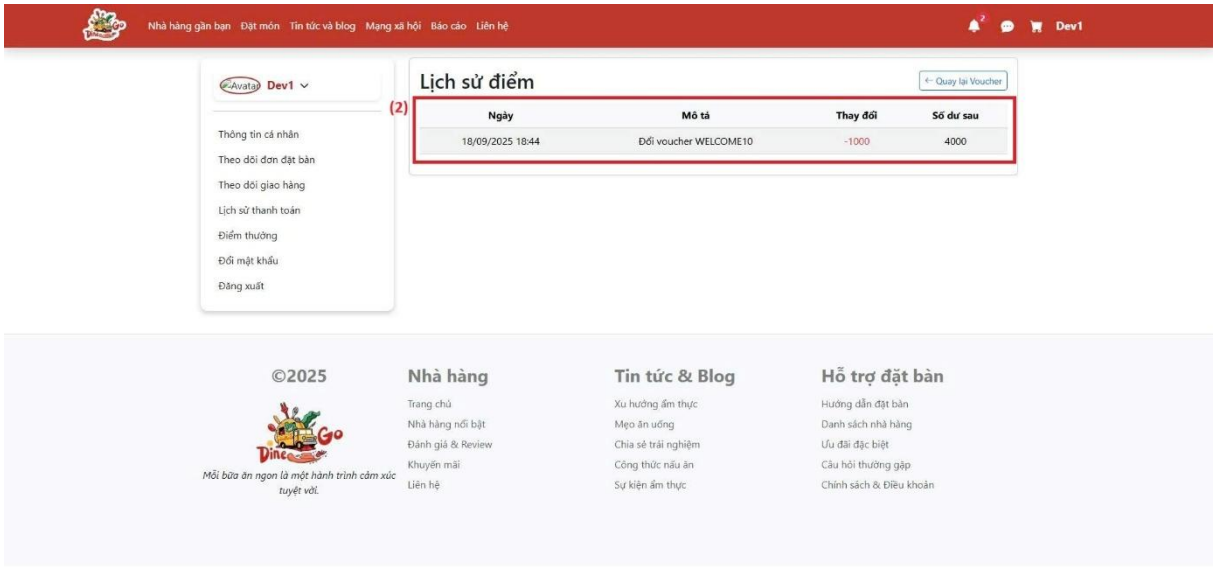


Figure 391 : Point transaction history page

- (2) Displaying all point transactions

3.3.3 Restaurant Owner feature

a. Create restaurant profile

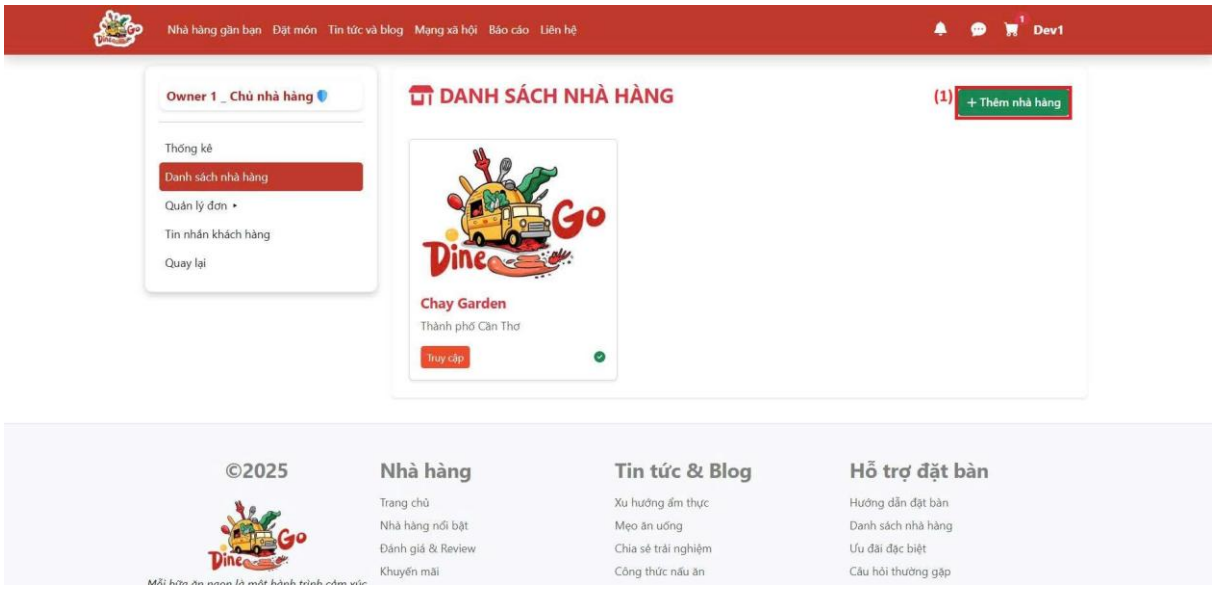


Figure 392: Create restaurant profile feature

This feature allows the Restaurant Owner to create a restaurant profile

- (1) [+ Thêm nhà hàng] button to open the create restaurant form.

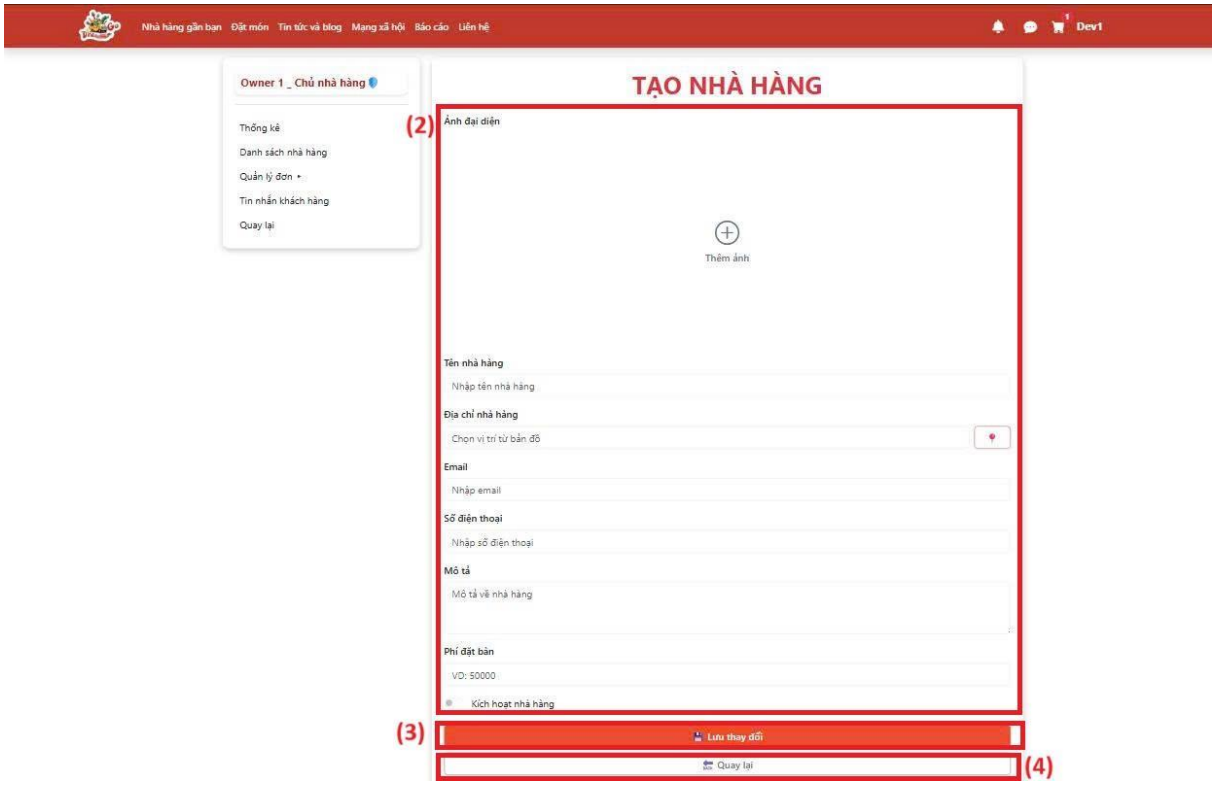


Figure 393: Create restaurant profile form

- (2) Fill all fields
- (3) [Lưu thay đổi] button to save the restaurant profile.
- (4) [Quay lại] button to return to the restaurant list without saving.
    - "Tên nhà hang" is required and must be unique.
    - "Địa chỉ nhà hang" is required.

- "Email" must be in a valid email format.
- "Số điện thoại" must contain only digits and follow valid phone format.
- "Phí đặt bàn" must be a positive numeric value.
- "Ảnh đại diện" (if uploaded) must be in JPG/PNG format and within the allowed size.
- If creation is successful, the new restaurant profile is added to the list.
- If creation fails, an error message is displayed.

b. Update restaurant information

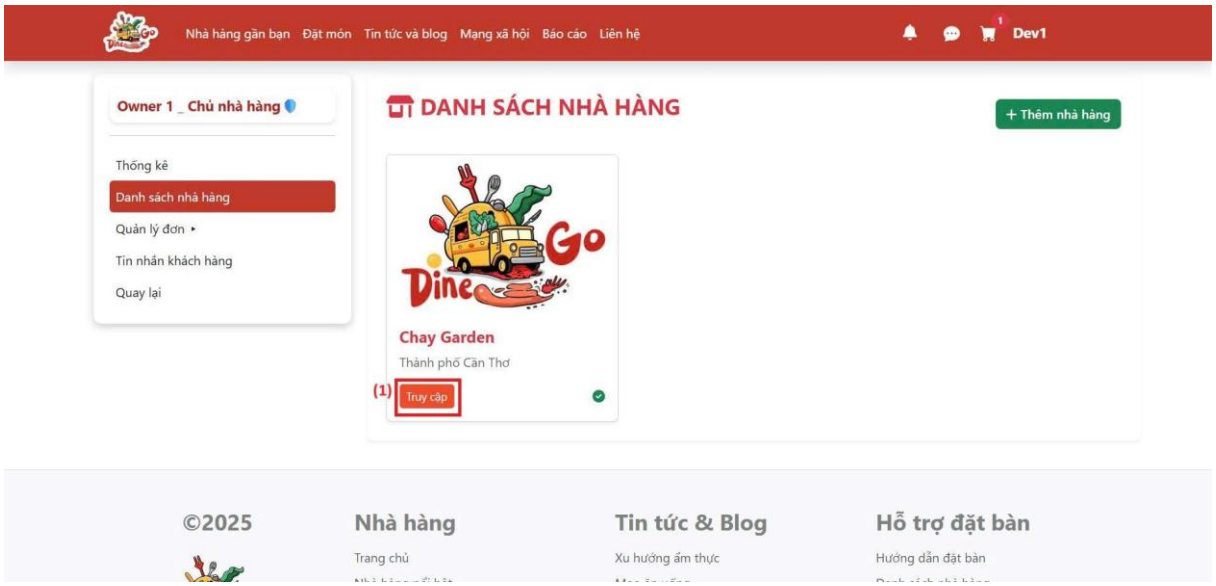


Figure 394: Update restaurant feature

This feature allows the Restaurant Owner to update their restaurant's information.

- (1) [Truy cập] button to open the restaurant detail page.

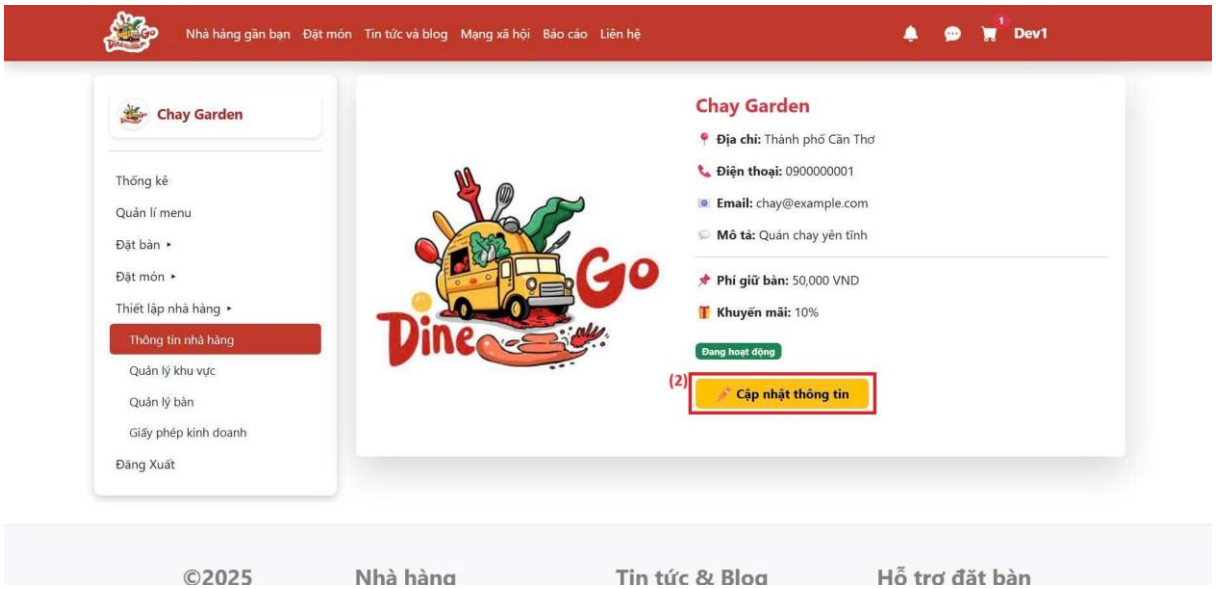


Figure 395: Update restaurant profile feature

- (2) [Cập nhật thông tin] button to open the update restaurant form.

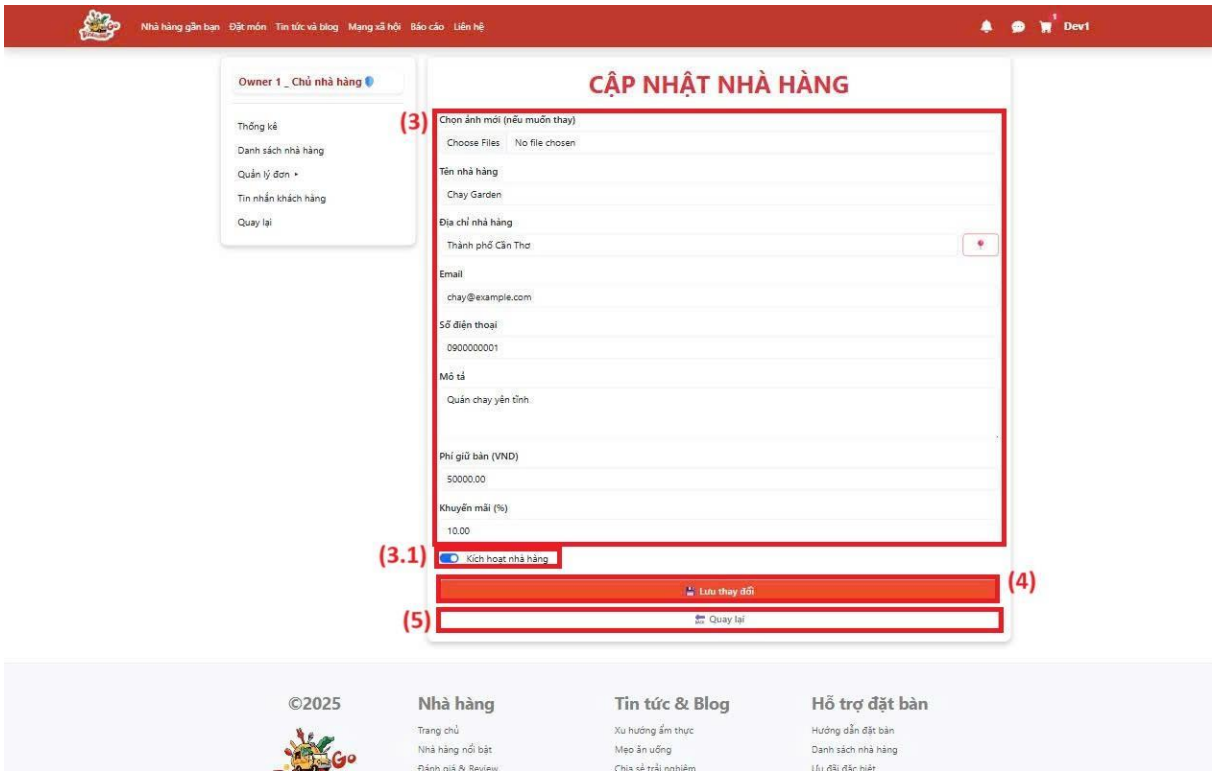


Figure 396: Update restaurant profile feature

- (3) Edit form fields want to edit
- (3.1) Kích hoạt nhà hàng – Toggle to enable or disable the restaurant. If not activated, the restaurant will be marked as closed.
- (4) [Lưu thay đổi] button to save the updated information.
- (5) [Quay lại] button to return to the restaurant detail page without saving.
    - "Tên nhà hang" is required and must be unique.
    - "Địa chỉ nhà hang" is required.
    - "Email is required and must follow valid email format.
    - "Số điện thoại" is required, must contain only numbers, and must meet length requirements.
    - "Phí giữ bàn" is required and must be a numeric value.
    - If the update is successful, the system displays the updated restaurant profile. If it fails, an error message is shown.

c. Manage menu

iv. View menu

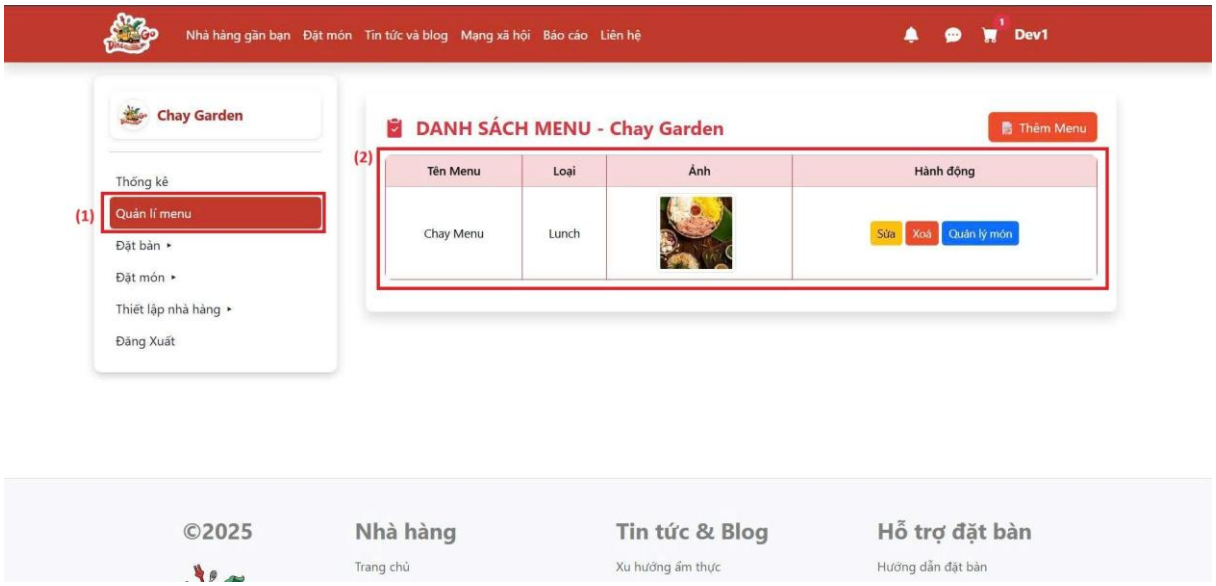


Figure 397: Menu page of DineGO

This feature allows the Restaurant Owner to view and manage all menus of their restaurant.

- (1) [Quản lí menu] – Menu navigation option to open the menu management page.
- (2) Menu list table – Displays all existing menus of the restaurant with the following columns
  - "Loại menu" must be selected from available options.
  - If an image is uploaded, it must be in JPG/PNG format and within the allowed file size.
  - Deletion of a menu is only allowed if it has no active orders linked to its foods.

v. Add menu

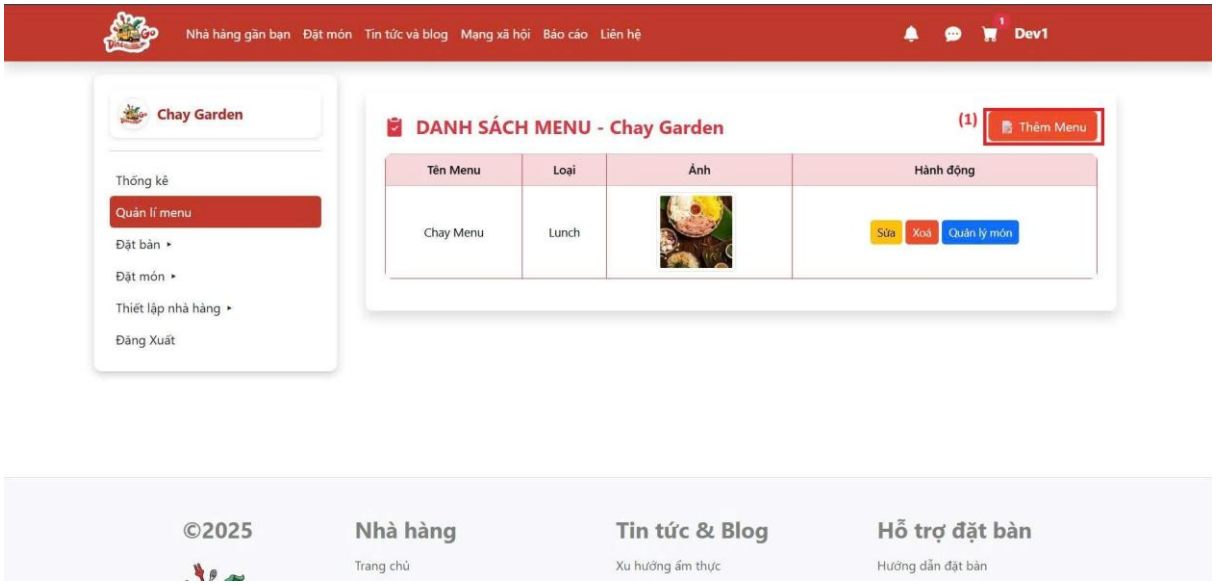


Figure 398: Menu page of DineGO

This feature allows the Restaurant Owner to create a new menu for the selected restaurant.

- (1) [Thêm Menu] button to open the add menu form.

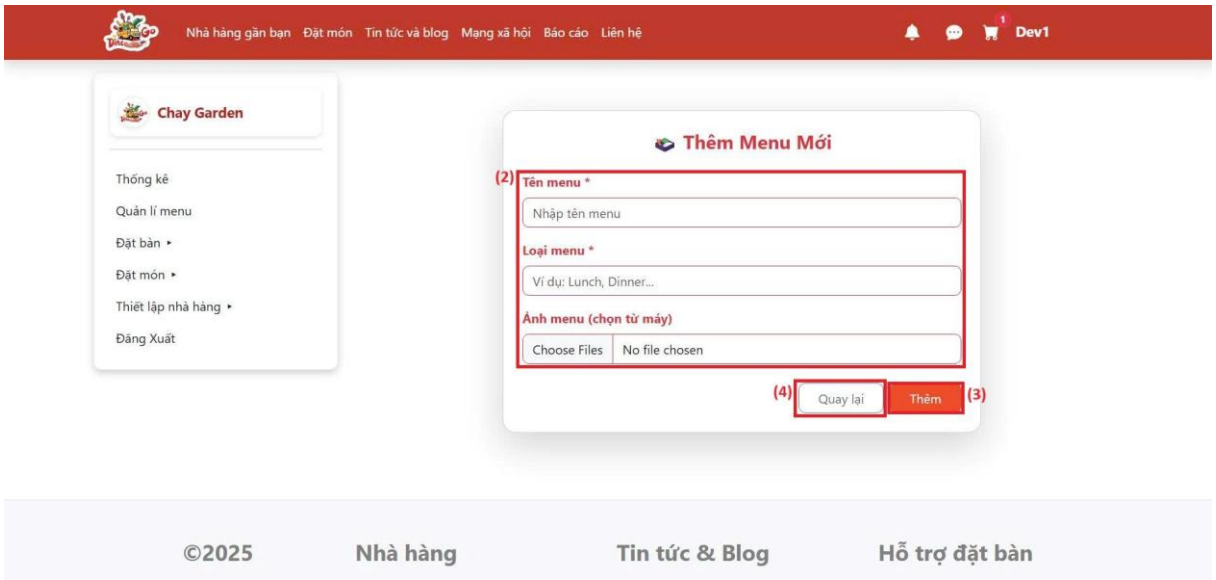


Figure 399: Menu page of DineGO

- (2) Fill all fields
- (3) [Thêm] button to save the new menu
- (4) [Quay lại] button to return to the menu list without saving.
    - "Tên menu" is required and must be unique within the restaurant
    - "Loại menu" is required and should not be empty
    - "Ảnh menu" must be in JPG/PNG format and within the allowed size
    - If creation succeeds, the new menu appears in the list; if it fails, an error message is displayed.

vi. Update menu

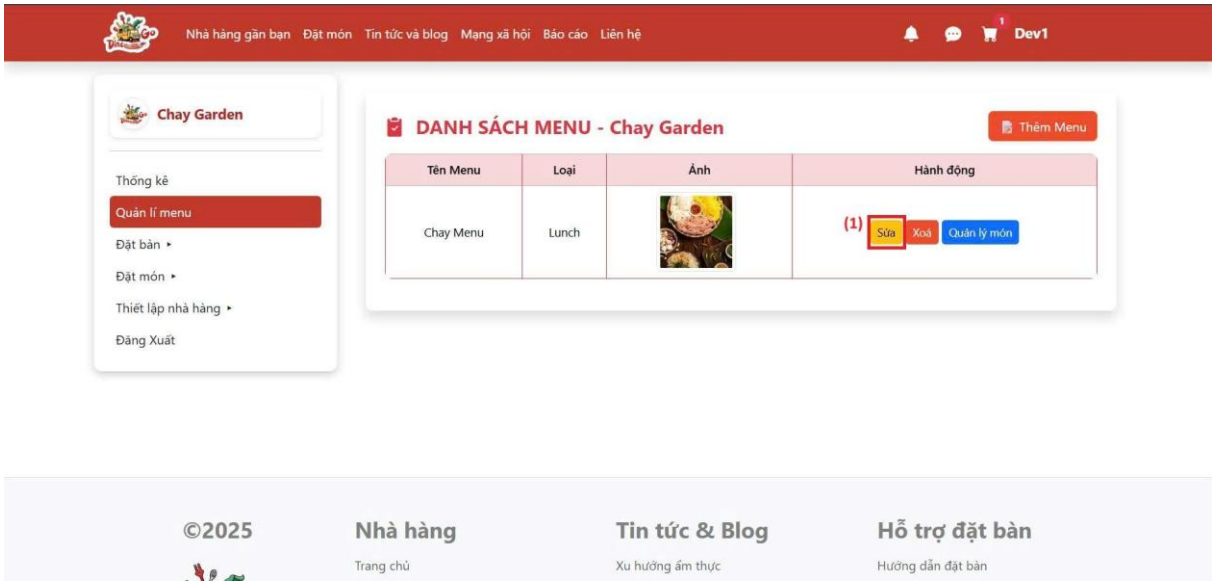


Figure 400: Menu page of DineGO

This feature allows the Restaurant Owner to update the existing menu information.

- (1) [Sửa] button to open the edit menu form.

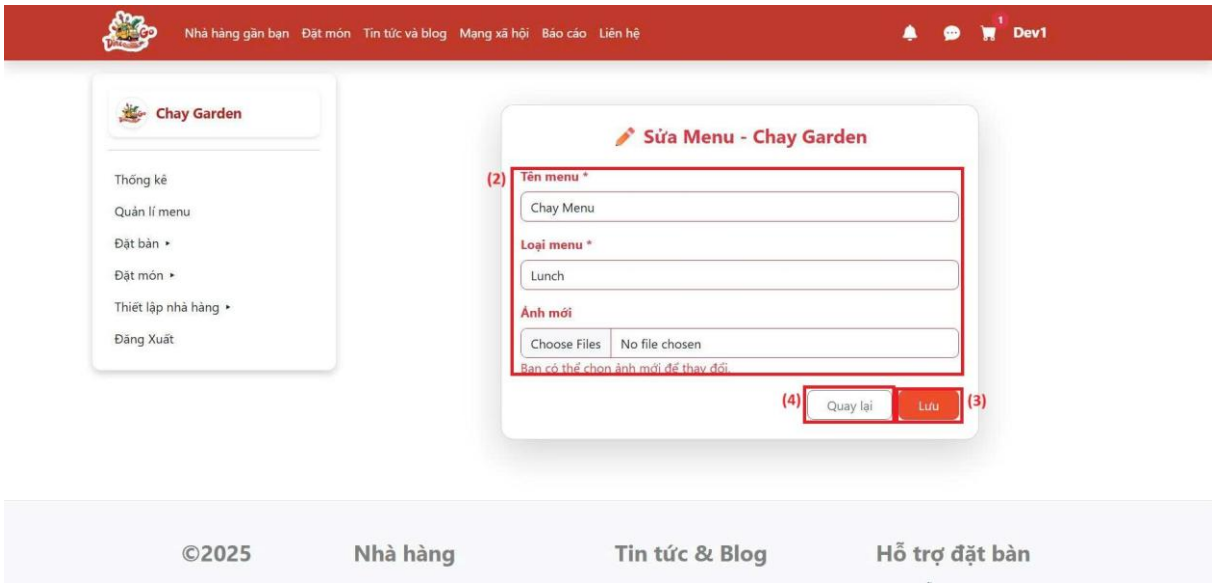


Figure 401: Menu page of DineGO

- (2) Edit form fields want to edit
- (3) [Lưu] button to save the updated menu information.
- (4) [Quay lại] button to return to the menu list without saving changes.
    - "Tên menu" is required and must be unique within the restaurant.
    - "Loại menu" is a required selection.
    - On success, the updated menu information will be displayed in the list; on failure, an error message will appear.

vii. Delete menu

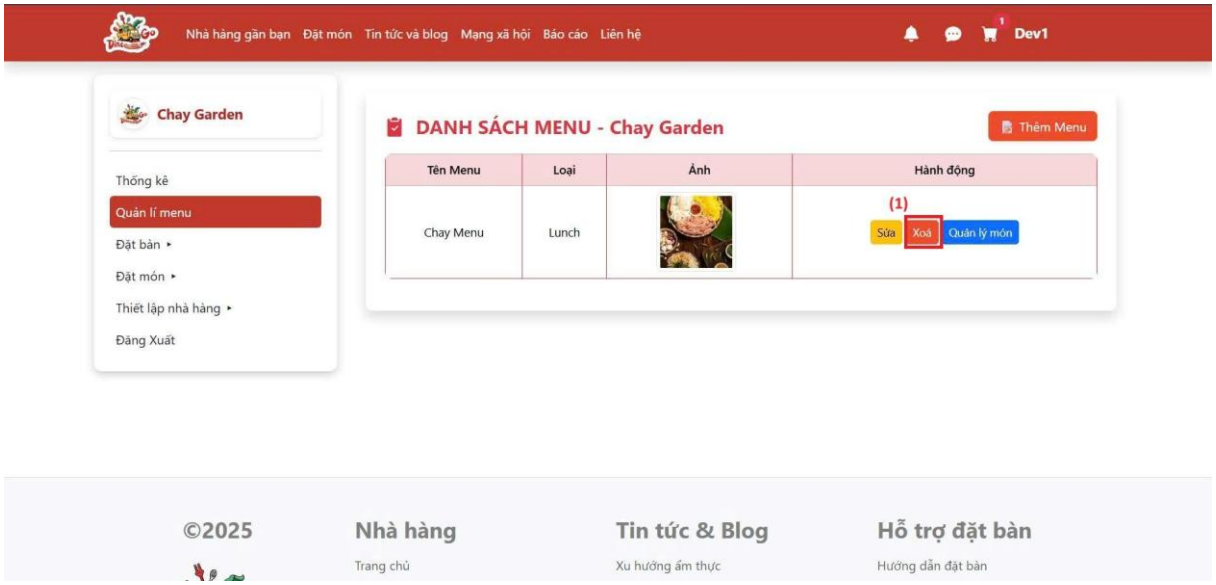


Figure 402: Menu page of DineGO

This feature allows the Restaurant Owner to remove an existing menu

- (1) [Xoá] button to open the delete confirmation screen.

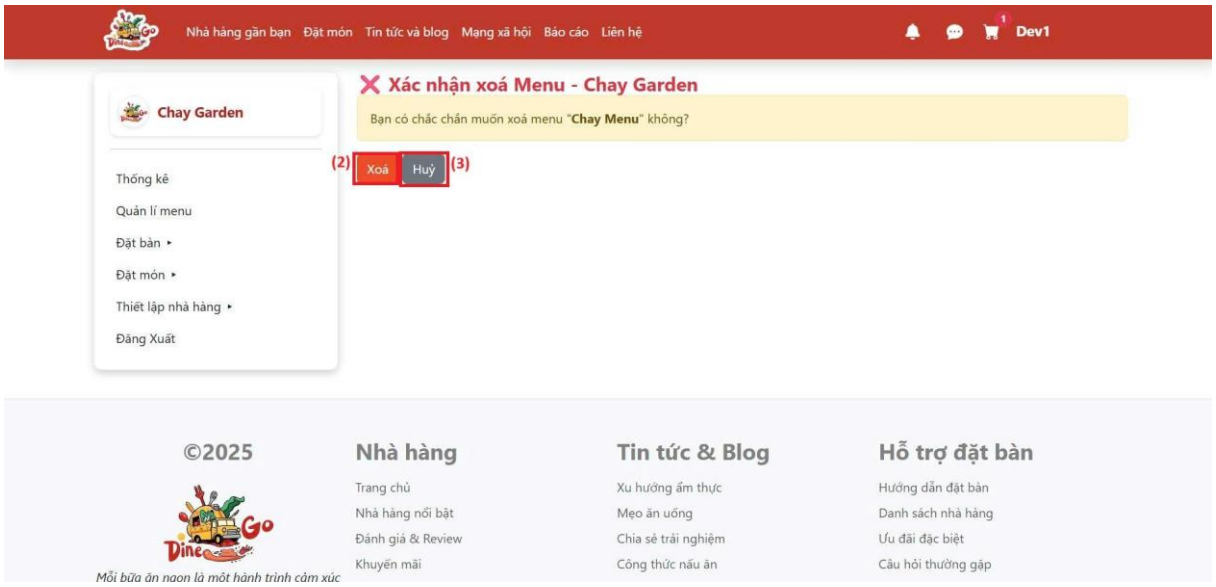


Figure 403: Confirm delete menu

- (2) [Xoá] button in the confirmation screen to permanently delete the selected menu.
- (3) [Huỷ] button to cancel the delete action and return to the menu list.
    - Deleting a menu will remove it permanently from the system and cannot be undone.
    - If the menu contains food items, the system may prompt for confirmation before deletion.
    - On success, the deleted menu will no longer appear in the list

d. Management food

i. View food

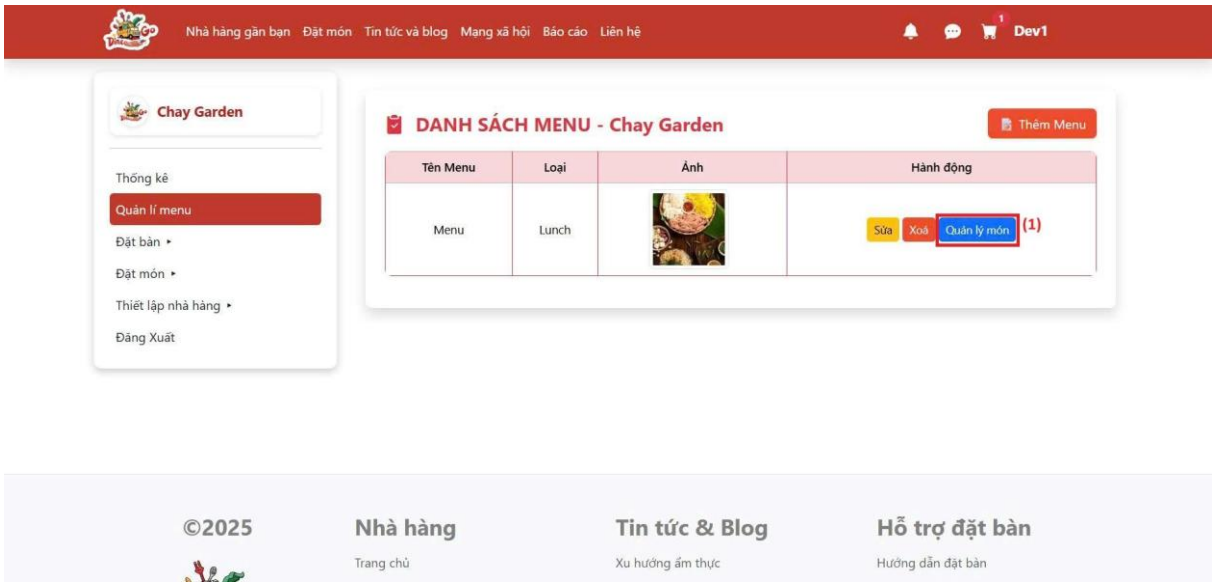


Figure 404: Menu page of DineGO

This feature allows the Restaurant Owner to view the list of food items within a specific menu.

- (1) [Quản lý món] button – Opens the food management page for the selected menu.

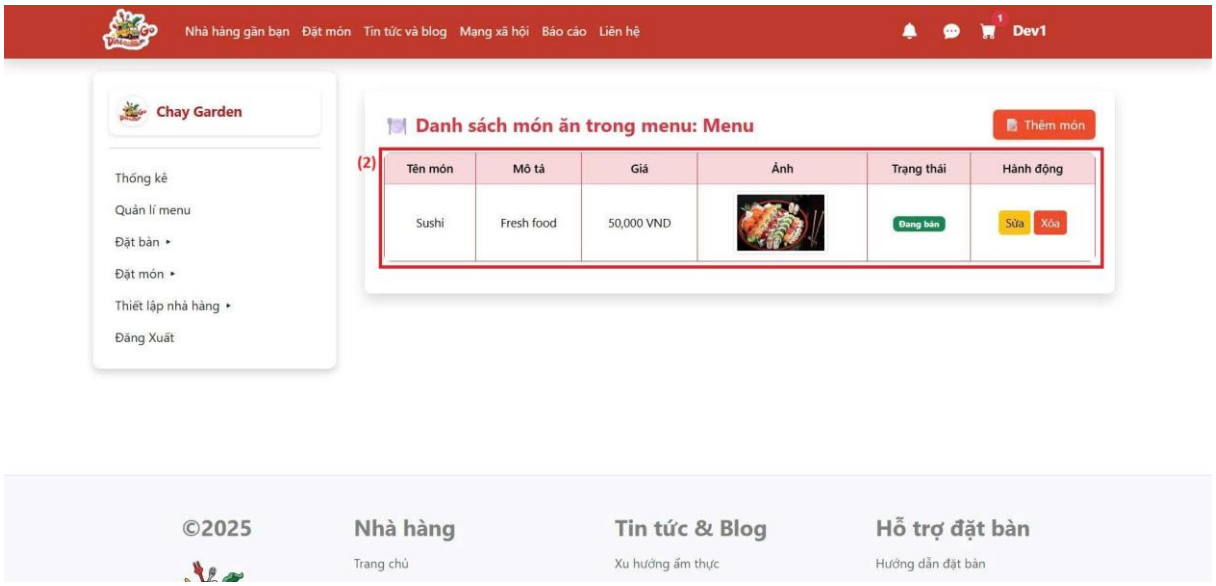


Figure 405: View food in menu

- (2) Displays all food items in the selected menu
  - Food information must be accurate and complete.
  - Images should be in accepted formats and within the size limit.
  - "Trạng thái" must correctly reflect the actual selling condition of the food

ii. Add food

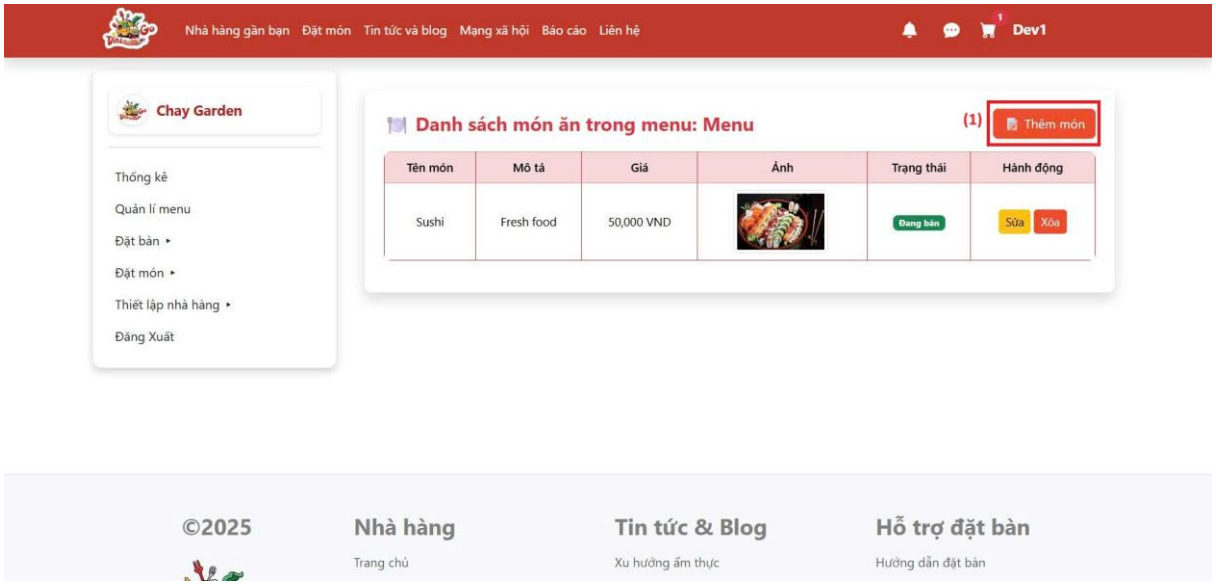


Figure 406: Food page in menu

This feature allows the Restaurant Owner to add a new food item into a selected menu.

- (1) [Thêm món] button – Opens the add food form.

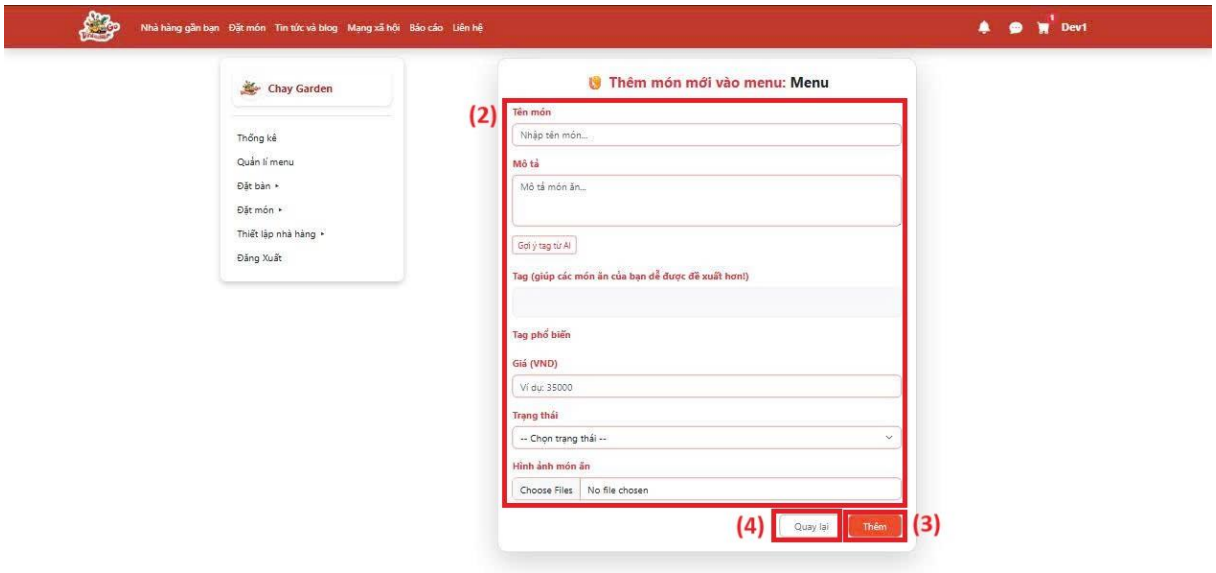


Figure 407: Add food feature

- (2) Fill all fields
- (3) [Thêm] button to save the new food item to the menu.
- (4) [Quay lại] button – Returns to the food list without saving.
    - "Tên món" is required
    - "Mô tả" is required.
    - "Giá" must be a valid number and ≥ 0.
    - "Trạng thái" is required.
    - Images must be in accepted formats and within the configured size limit.
    - On success, the new food item will appear in the food list; on failure, an error message will be shown

iii. Update food

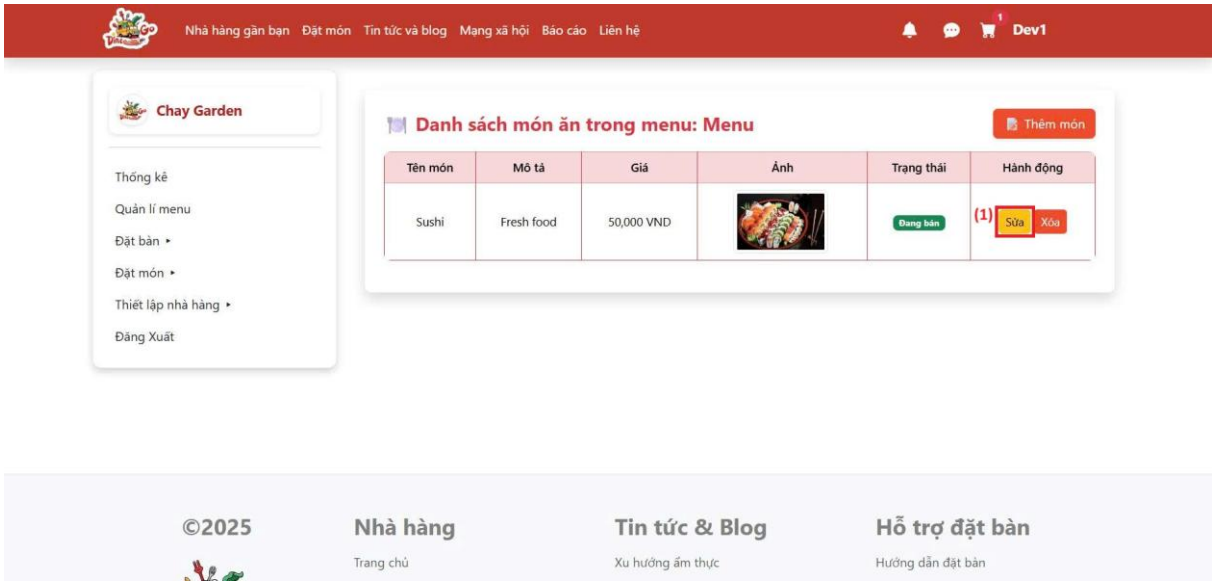


Figure 408: View food in menu

This feature allows the Restaurant Owner to update detail of an existing food item.

- (1) [Sửa] button to opens the edit food form for the selected dish.

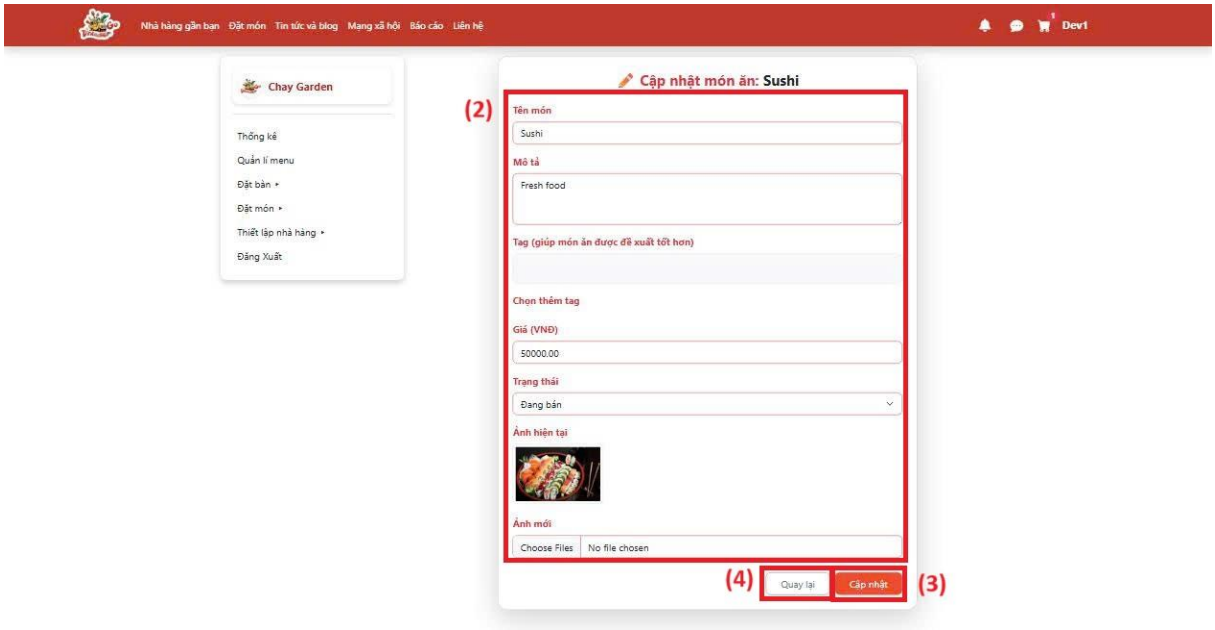


Figure 409: Update food feature

- (2) Edit form fields want to edit
- (3) [Cập nhật] button to saves the updated food detail.
- (4) [Quay lại] button returns to the food list without saving changes.
    - "Tên món" is required
    - "Mô tả" is required.
    - "Giá" must be a valid number and ≥ 0.
    - "Trạng thái" is required.
    - Images must be in accepted formats and within the configured size limit.
    - On success, the updated food item will be displayed in the food list; on failure, an error message will appear.

iv. Delete food

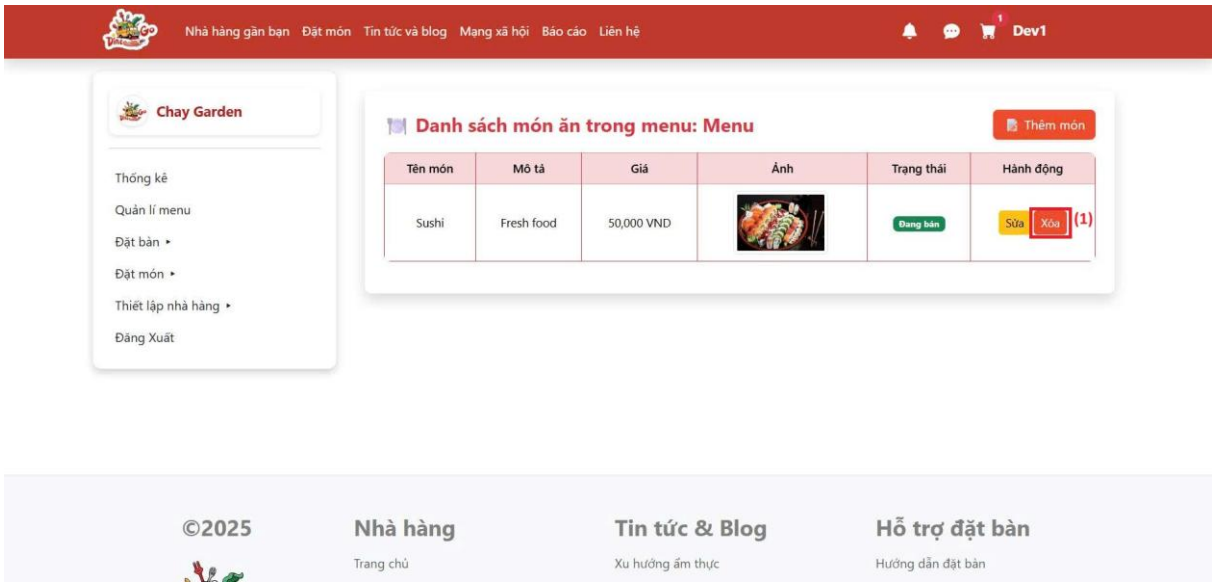


Figure 410: View food in menu

This feature allows the Restaurant Owner to remove an existing food item from a menu.

- (1) [Xoá] button to opens the delete confirmation screen for the selected dish.

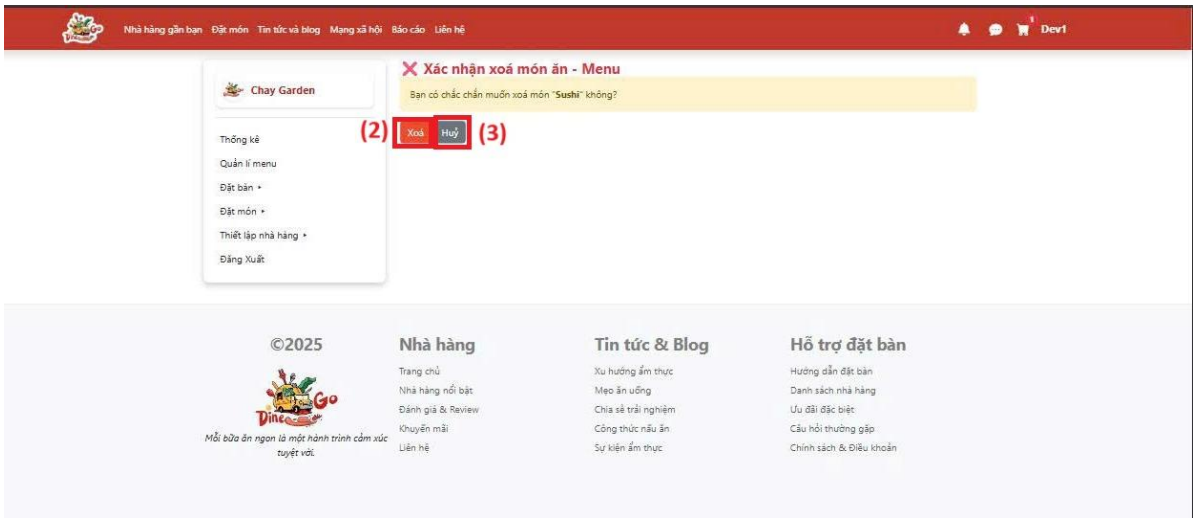


Figure 411: Delete food feature

- (2) [Xoá] button in the confirmation screen
- (3) [Huỷ] button to cancels the delete action and returns to the food list.
    - Deleting a food item will permanently remove it from the menu and cannot be undone.
    - If the food item is linked to existing orders, the system may require confirmation before deletion.
    - On success, the deleted food item will be removed from the list; on failure, an error message will be displayed.

e. Handle reservation

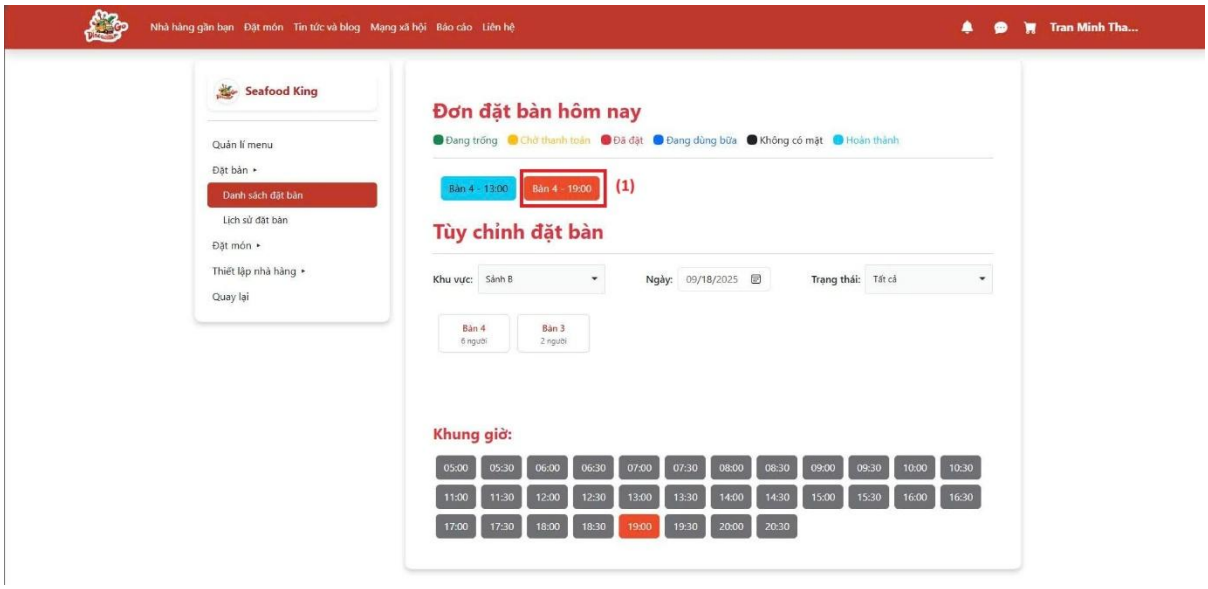


Figure 412: Handle reservation feature

This feature allows the Restaurant Owner to manage table reservations.

- (1) Select the table want to handle in section.

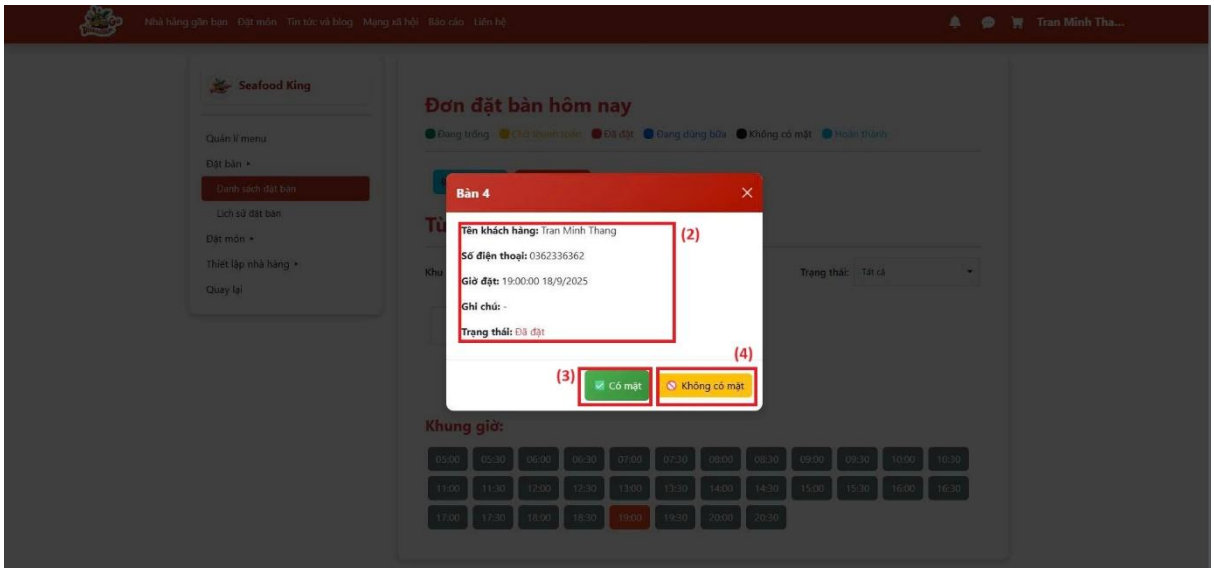


Figure 413: Detail table

- (2) Display detail table
- (3) [Có mặt] button to update tale status change to “Có mặt”
- (4) [Không có mặt] button to update table status change to “Không có mặt”.

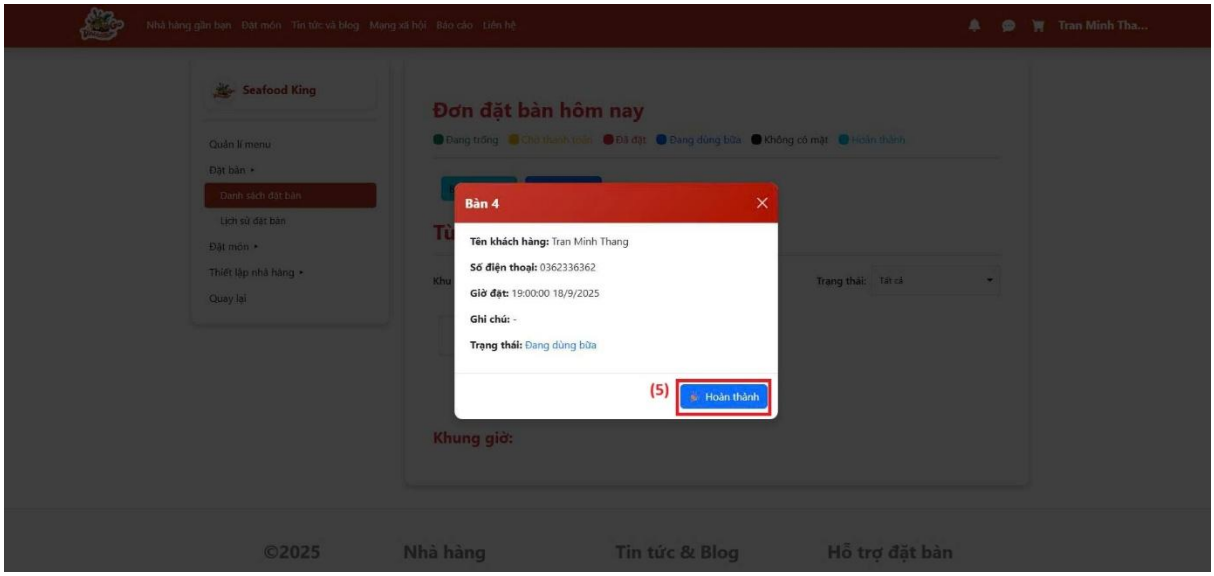


Figure 414: Handle reservation feature

- (5) [Hoàn thành] button to update table status to “Hoàn thành”
  - Only tables in available status can be booked.
  - On success, the table status will be updated in real time; on failure, an error message will be shown
  - Only change the state of "Có mặt" to "Hoàn thành"

f. Process food order

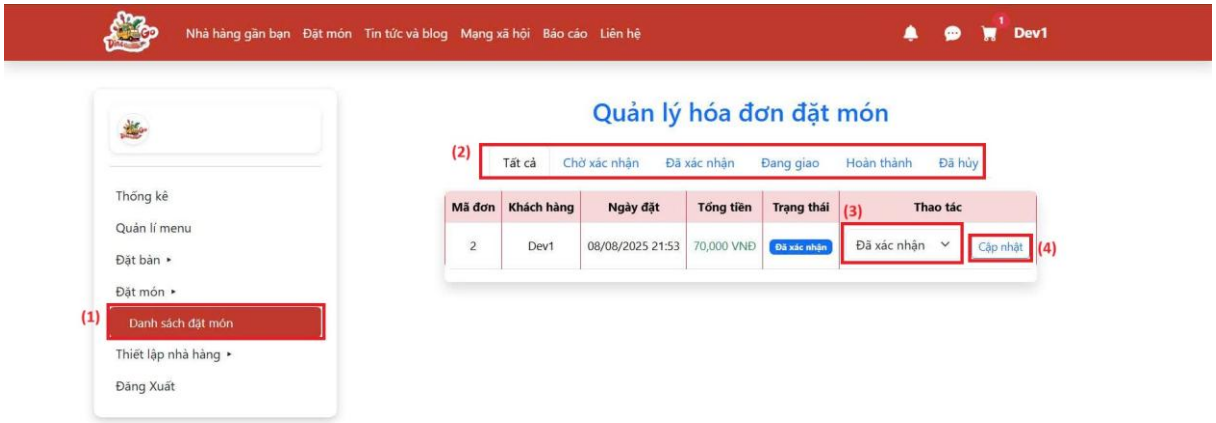


Figure 415: Process food order feature

This feature allows the Restaurant Owner to view and update the status of customer food orders.

- (1) [Danh sách đặt món] to opens the order management page.
- (2) Order status filter
- (3) Status dropdown to select a new status for the order.
- (4) [Cập nhật] button to save the new status for the selected order.
    - Orders must be updated to the correct status following the restaurant's workflow.
    - Status changes should reflect the actual progress of the order.
    - On success, the order list will refresh to show the updated status; on failure, an error message will appear.

g. Respond to customer inquiries

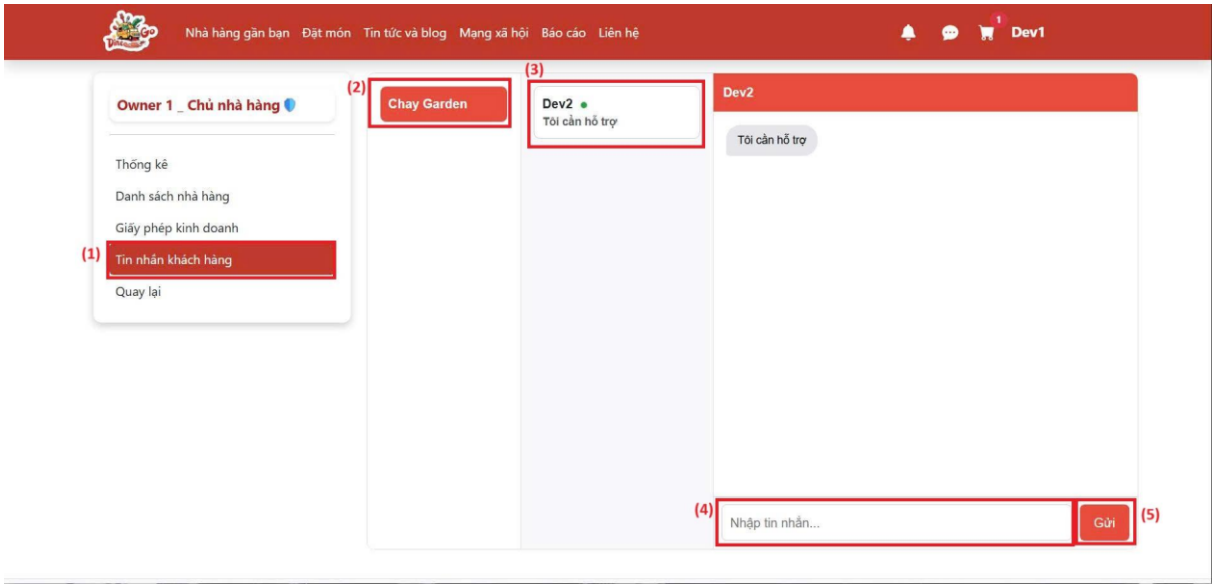


Figure 416: Respond to customer inquiries feature in DineGO

This feature allows the Restaurant Owner to chat directly with customers.

- (1) [Tin nhắn khách hàng] menu to open the customer message list page.
- (2) Select the restaurant to view its customer chats.
- (3) Shows the list of customers who have messaged the selected restaurant, with their latest message and online status.
- (4) Enter your message
- (5) [Gửi] button to send the message to the customer.
    - A restaurant must be selected before viewing messages.
    - Message content cannot be empty before sending.
    - Only active chats for the selected restaurant will be displayed.

h. View statistics dashboard

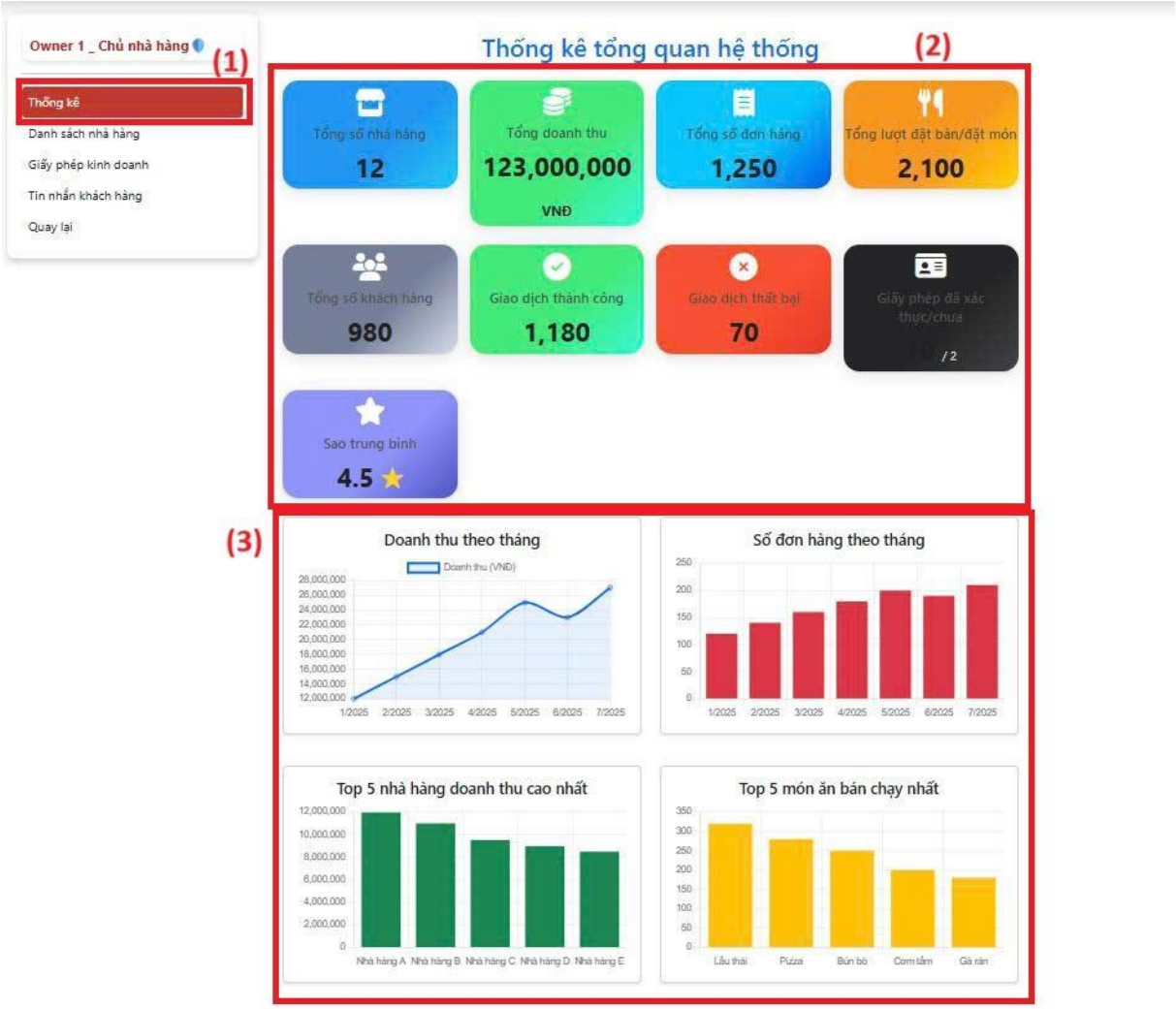


Figure 417: View statics dashboard in DineGO

This feature allows the Restaurant Owner to view the system's overall business performance and key statistics.

- (1) [Thống kê] menu – Click to open the statistics dashboard.
- (2) Display total statistics
- (3) Charts & rankings

i. View business license

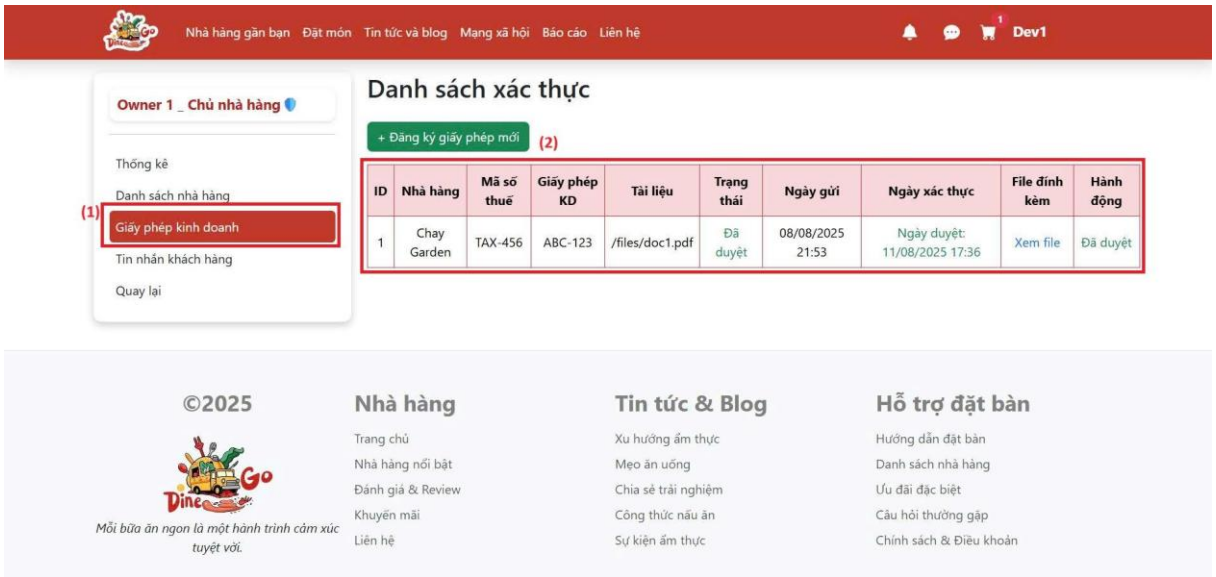


Figure 418: View business license in DineGO

This feature allows the Restaurant Owner to view the status and detail of their submitted business license.

- (1) [Giấy phép kinh doanh] – Opens the business license management page.
- (2) Displays all submitted business licenses with detail
    - Only valid and complete documents will be approved.
    - Uploaded files must be in the accepted formats and within the size limit.
    - File content must match the registered business detail.

j. Submit business license

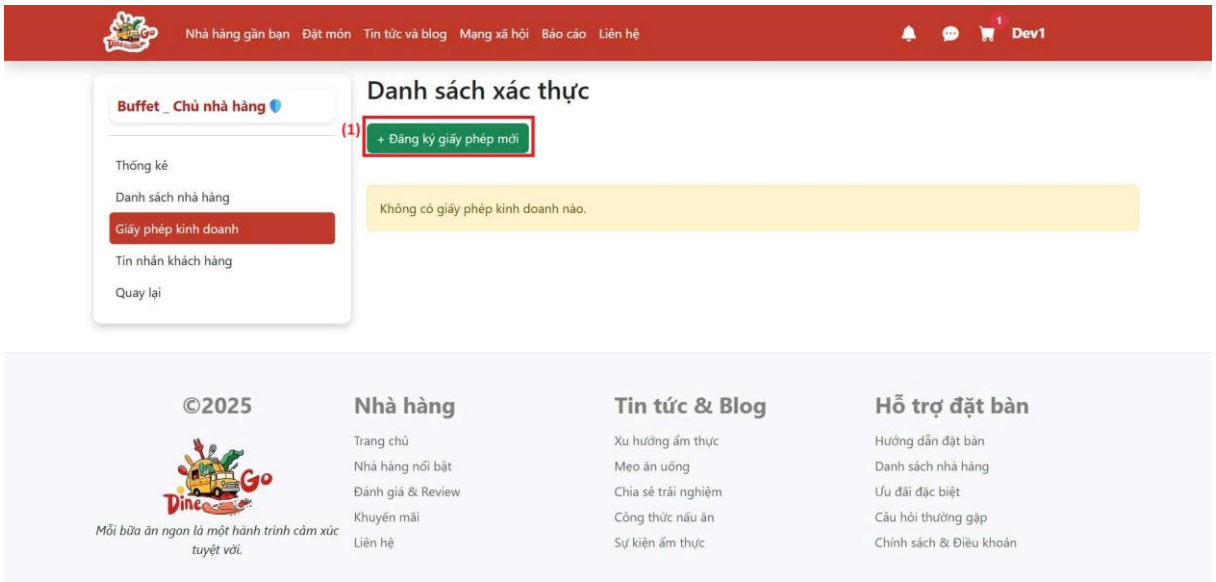


Figure 419: Submit business license feature in DineGO

This feature allows the Restaurant Owner to submit a new business license for approval.

- (1) [+ Đăng ký giấy phép mới] button to open the registration form.

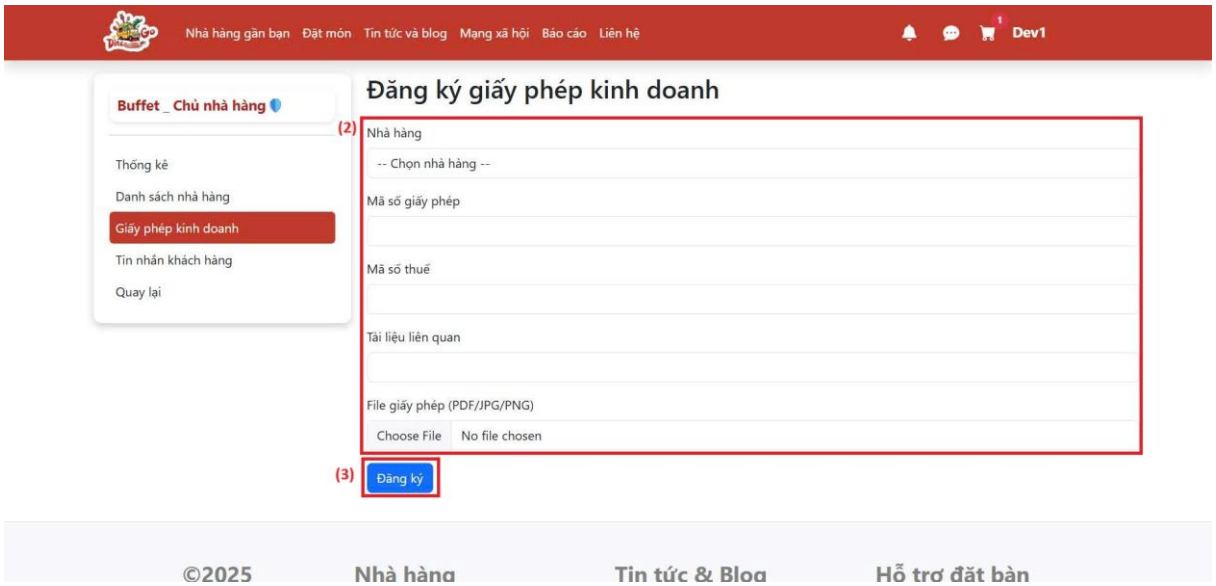


Figure 420: Submit business license feature in DineGO

- (2) Fill all fields
- (3) [Đăng ký] button to submit the license for review.
    - "Nhà hàng" must be selected.
    - "Mã số giấy phép" is required and must follow license format.
    - "Mã số thuế"s is required and must follow tax code format.
    - "File giấy phép" must be in PDF, JPG, or PNG format.

VII. Appendix

[1] DineGO Research Team, *DineGODevelopment_Task_Management.xlsx*, supplementary file included in submission package at: ./Final Thesis + Reports/ DineGODevelopment_Task_Management.xlsx, Aug. 2025.

[2] DineGO Database, *dinego_db.jpg*, supplementary file included in submission package at: ./Others/dinego_db.jpg, Aug. 2025.

[3] Dataset train AI food descriptions, *food_descriptions_20k.csv*, supplementary file included in submission package at: ./Others/food_descriptions_20k.csv, Aug. 2025.

[4] Dataset train AI prompt response, * prompt_response_15k_concrete.csv *, supplementary file included in submission package at: ./Others/prompt_response_15k_concrete.csv, Aug. 2025.

[3] Report 5 – Unit Test, * Report5_Unit Test.xlsx *, supplementary file included in submission package at: ./ Final Thesis + Reports/ Report5_Unit Test.xlsx, Aug. 2025.

[4] Report 5 – Test Report, * Report5_Test Report.xlsx *, supplementary file included in submission package at: ./ Final Thesis + Reports/ Report5_Test Report.xlsx, Aug. 2025.]