Team L

CCT Express
Software Requirements Specification
For AI-enabled online restaurant order and delivery system

Version 2.0

Revision History

Date	Version	Description	Author
22/10/25	1.0	Initial Specifications	Team L
17/11/25	2.0	Initial Specifications	Team L

Table of Contents

Software Requirements Specification

1. Introduction

1.1 Purpose

This document provides a detailed Software Requirements Specification for an AI-enabled online restaurant order and delivery system for CCT Express. The system is an intelligent restaurant management platform that integrates Large Language Model (LLM) technology to provide menu management, customer ordering, delivery services, and AI-powered customer service. This document comprehensively describes the external behavior, functional requirements, non-functional requirements, design constraints, and other necessary factors to provide a complete and comprehensive description of the software requirements.

1.2 Scope

The restaurant management system includes the following major functional sub-system:

- Account Management Sub-System
- Menu and Dish Management Sub-System
- Order processing and Payment Sub-System
- Delivery Management Sub-System
- Knowledge Base & AI Customer Service Sub-System
- Reputation and Human Resources Management Sub-System

The system is developed based on the Django framework, uses SQLite database, and integrates Ollama local LLM model and OpenRouteService API for intelligent customer service and route planning.

1.3 Definitions, Acronyms, and Abbreviations

- LLM: Large Language Model
- API: Application Programming Interface
- VIP: Very Important Person
- AI: Artificial Intelligence
- ORM: Object-Relational Mapping
- REST: Representational State Transfer

1.4 References

- Django Official Documentation: https://docs.djangoproject.com/
- Ollama Documentation: https://ollama.ai/docs
- OpenRouteService API Documentation: https://openrouteservice.org/dev/
- Bootstrap 5 Documentation: https://getbootstrap.com/docs/5.1/
- Route Planning Setup Document: ROUTE_PLANNING_SETUP.md

1.5 Overview

The remainder of this document is organized as follows:

- Section 2 provides an overall description of the system, including use case model survey and assumptions and dependencies
- Section 3 details specific requirements, including use case reports and supplementary requirements

- Section 4 provides supporting information, including index and appendices

2. Overall Description

2.1 Use-Case Model Survey

The system supports three main user role categories:

2.1.1 User Role Categories

2.1.1.1 Employee Users

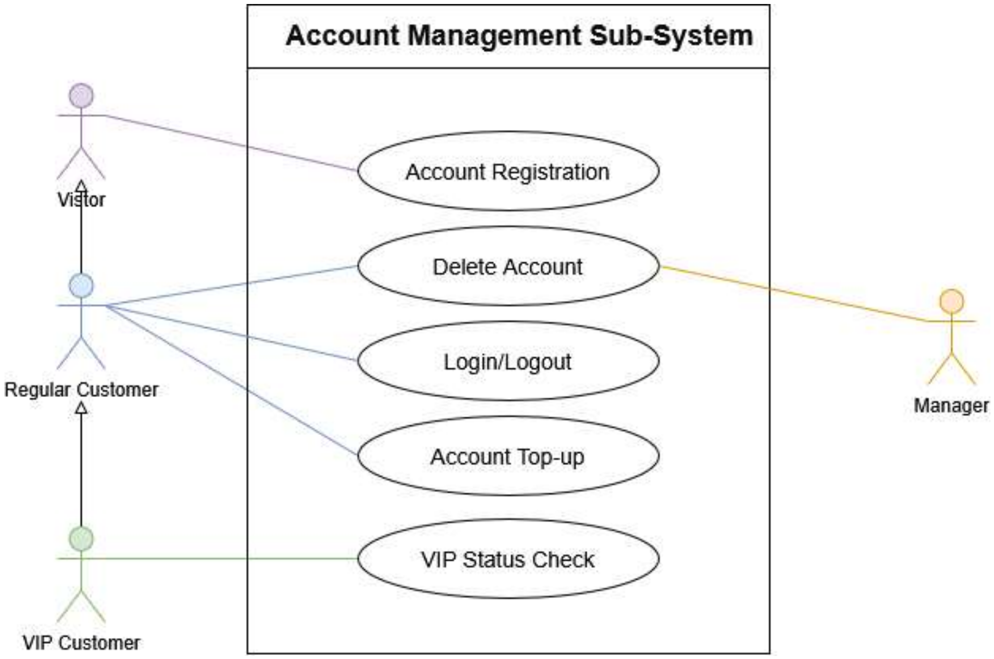
- **Manager**: Handles customer account, registration, reputation and human resources maintenance employee management, financial management
- **Chef**: Responsible for creating and managing dish menus, accepting order and preparing food
- **Delivery Person**: Responsible for food delivery, participating in delivery bidding

2.1.1.2 Customer Users

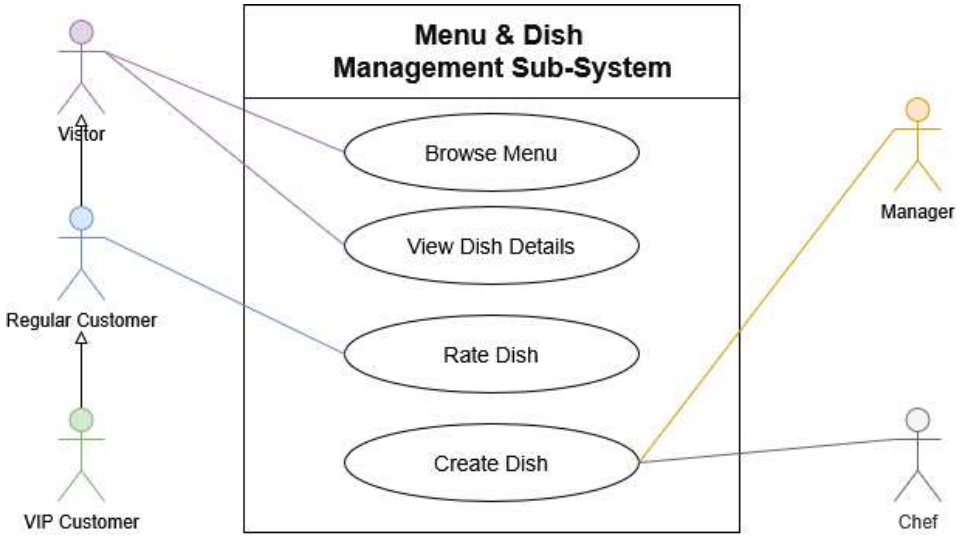
- **Regular Customer**: Registered customers who can browse, ask questions, order, rate dish and delivery services, and make contribution to the local knowledge base
- **VIP Customer**: Customers who have spent over $100 or completed 3 orders, enjoying 5% discount and free delivery
- **Visitor**: Can browse menus, ask questions, and apply for registration

2.1. 2 System Features

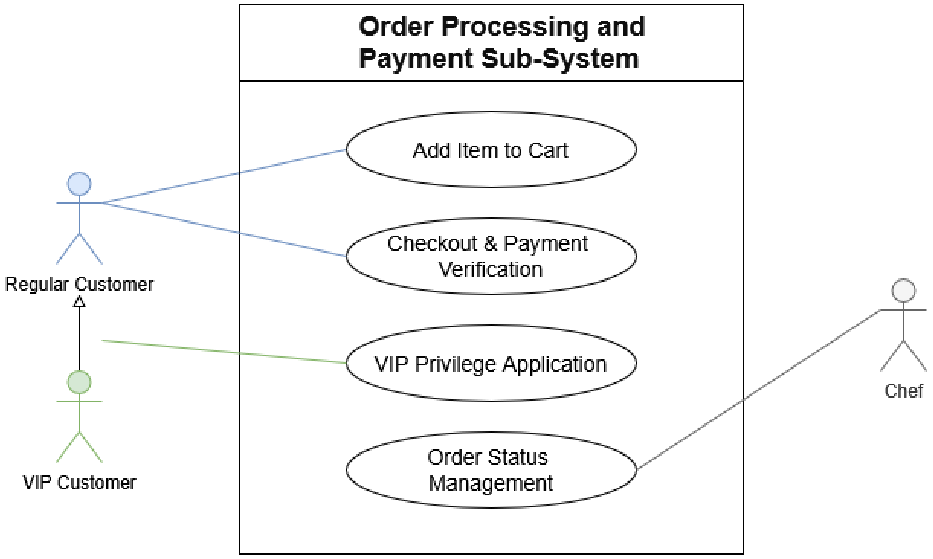
- **Account Management Sub-System:** Opened for registration, user login/out and basic account management including deleting account and top-up account wallet



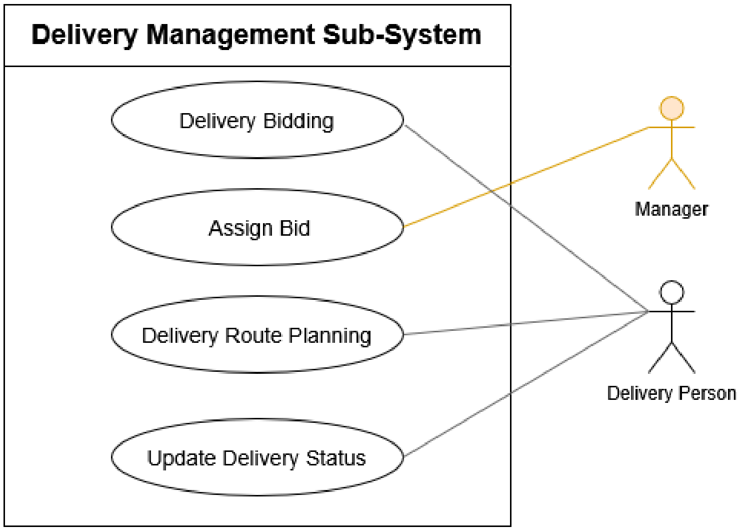
- **Menu and Dish Management Sub-System:** Basic menu and dish detail browsing, rating feature for registered customers, and dishes management by chef



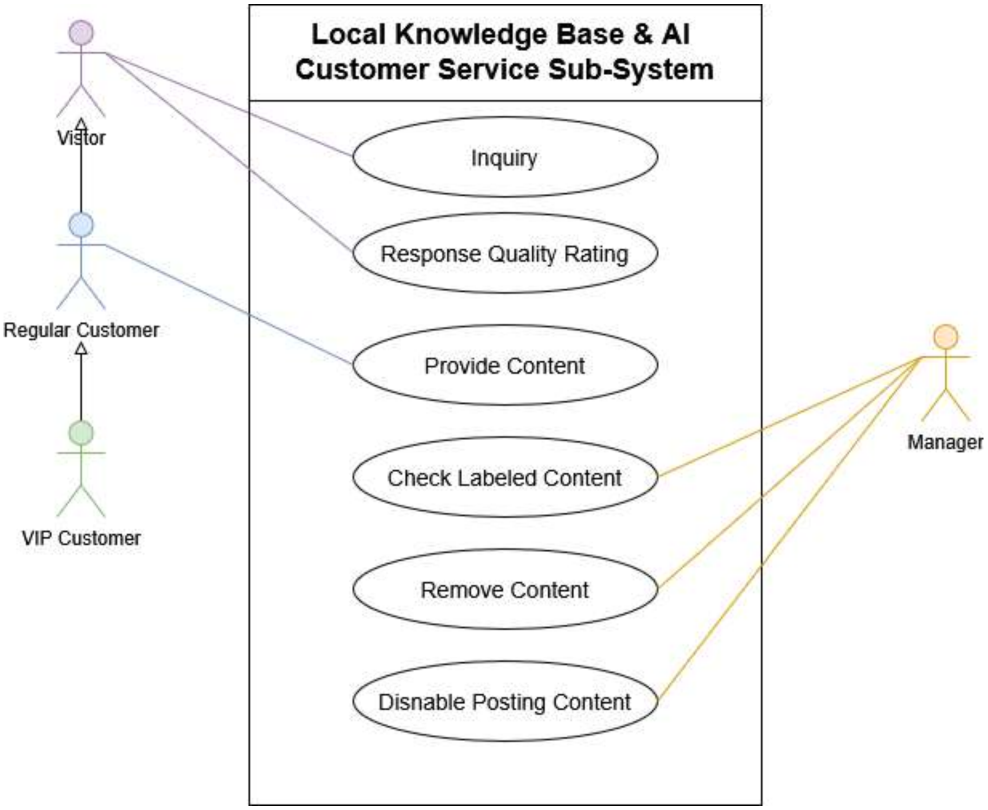
- **Order Processing Sub-System:** Includes shopping cart, order checkout, payment verification and order status management



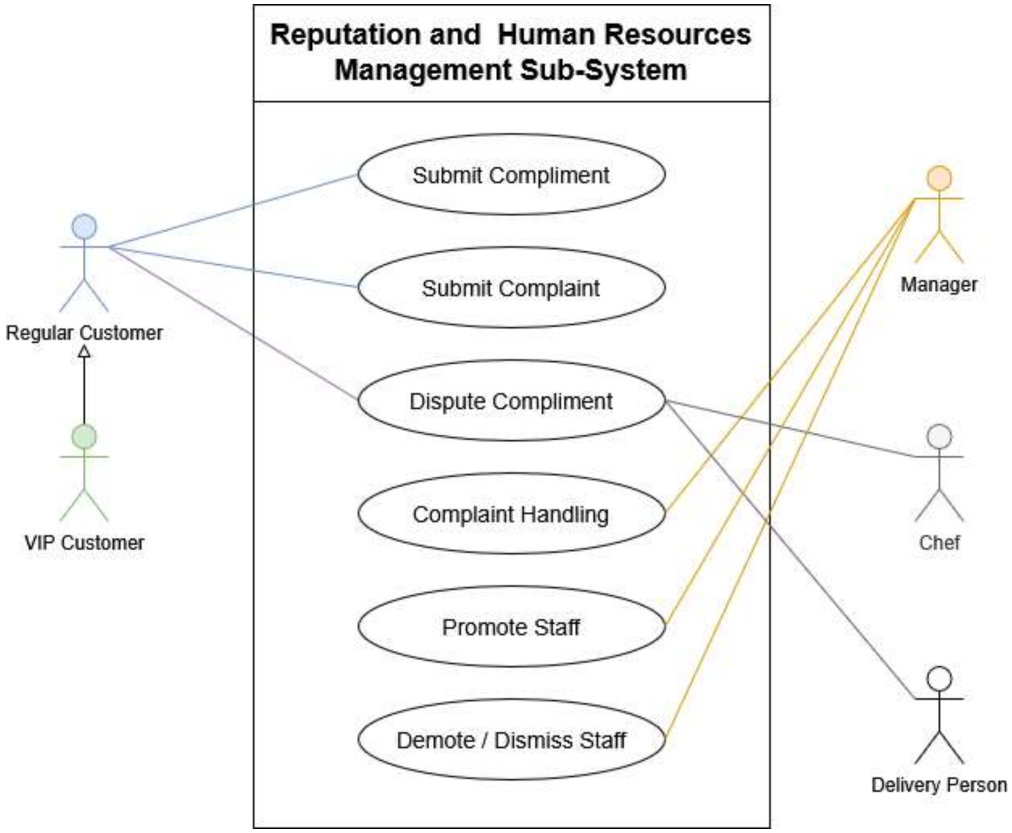
- **Delivery Management Sub-System:** Delivery person bidding, route planning, delivery tracking



- **Knowledge Base & AI Customer Service Sub-System:** Provides response to customers user's inquiry, content rating and local knowledge base management performed by manager



- **Reputation and Human Resources Management Sub-System:** Complaint and compliment handling, warning system, VIP management



2.2 Assumptions and Dependencies

2.1.2 Technical Assumptions

- System runs in an environment supporting Python 3.8+
- Ollama and llama3:8b model are installed *and configured[NOT YET!!]*
- Stable network connection with access to OpenRouteService API *waited to be tested[NOT YET!!]*
- User devices support modern web browsers

2.1.2 Business Assumptions

- Restaurant has stable network connection and computing resources
- Customers have basic network usage capabilities
- Delivery personnel are familiar with mobile device operations
- Managers have basic system management capabilities

2.1. 3 External Dependencies

- **Ollama Service**: Provides local LLM inference capabilities
- **OpenRouteService API**: Provides route planning and geocoding services
- **Django Framework**: Provides web application development foundation
- **SQLite Database**: Provides data persistence storage

3. Specific Requirements

3.1 Use-Case Reports

UC-001: Account Registration

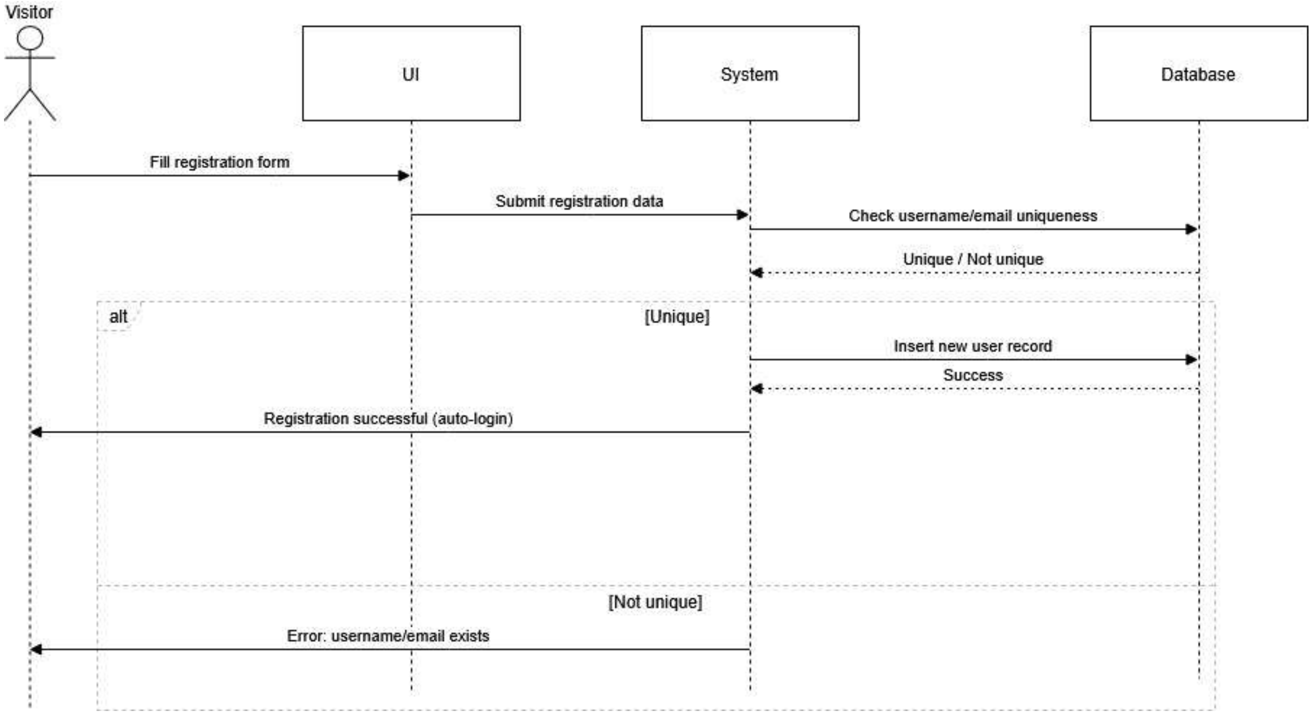
Description: New users register system accounts

Actors: Visitor, System

Main Flow:

1. User selects user type (customer/chef/delivery)
2. Fills in registration information (username, email, password, etc.)
3. System validates information format and uniqueness
4. Creates user account and corresponding role profile
5. Automatically logs in user

Exceptional Scenario: Username already exists, password doesn't meet requirements/invalid email format



UC-002: User Login

Description: Registered users log into the system

Actors: User, System

Main Flow:

1. User enters username and password
2. System validates credentials
3. Redirects to appropriate dashboard based on user role

Exceptional Scenario:

- Wrong password → System shows "Invalid credentials."

- Account disabled → System blocks login.
- DB query fails → Login aborted.

UC-003: User Logout

Description: User securely exits the system

Actors: User, System

Main Flow:

1. User clicks logout button
2. System clears session information
3. Redirects to homepage

UC-004: Account Top-up

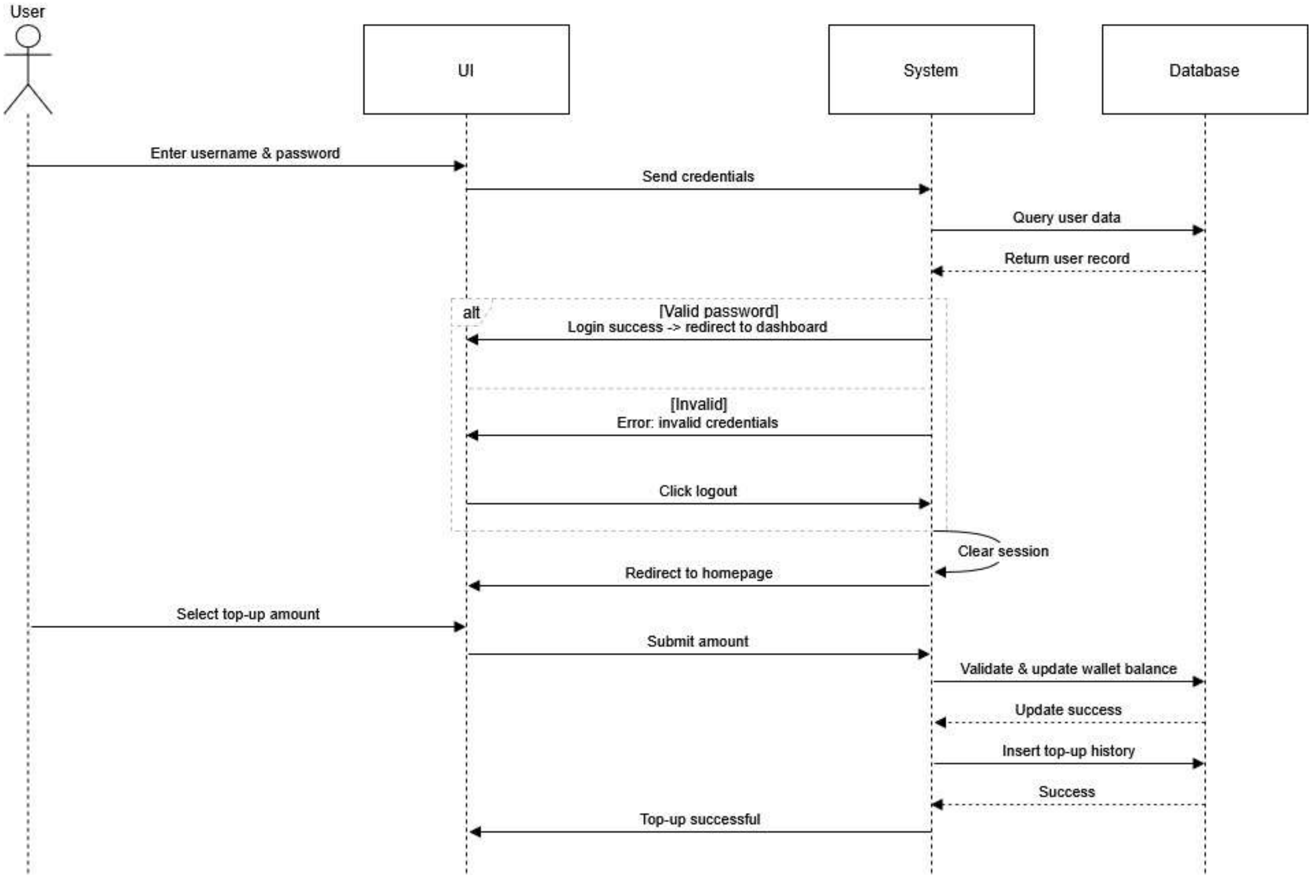
Description: Customer tops up account

Actors: Customer, System

Main Flow:

1. Customer selects top-up amount
2. System validates top-up information
3. Updates account balance
4. Records top-up history

Exceptional Scenario: Invalid amount (0 or negative).



UC-005: VIP Status Check

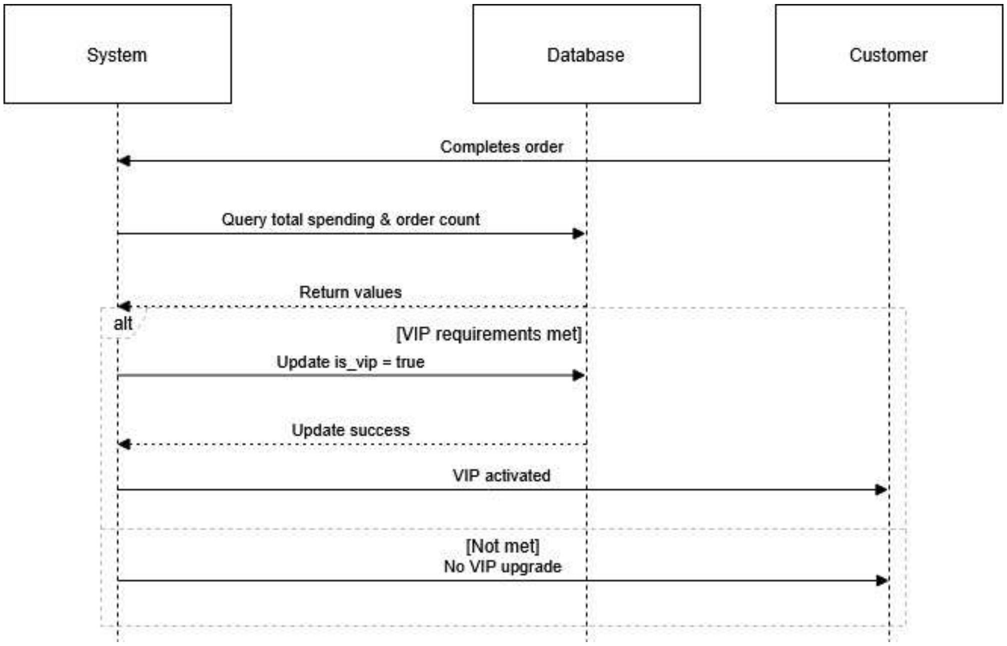
Description: System automatically checks customer VIP status

Actors: System

Trigger: Customer completes order

Main Flow:

1. Check if customer total spending ⩾ $100
2. Check if order count ⩾ 3
3. If conditions met, upgrade to VIP
4. Update customer status



UC-006: Delete Account

Description: System automatically checks customer VIP status

Actors: Customer, System, Manager

Trigger: Customer deregisters voluntarily; customer receives 3th warnings; manager dismisses staff

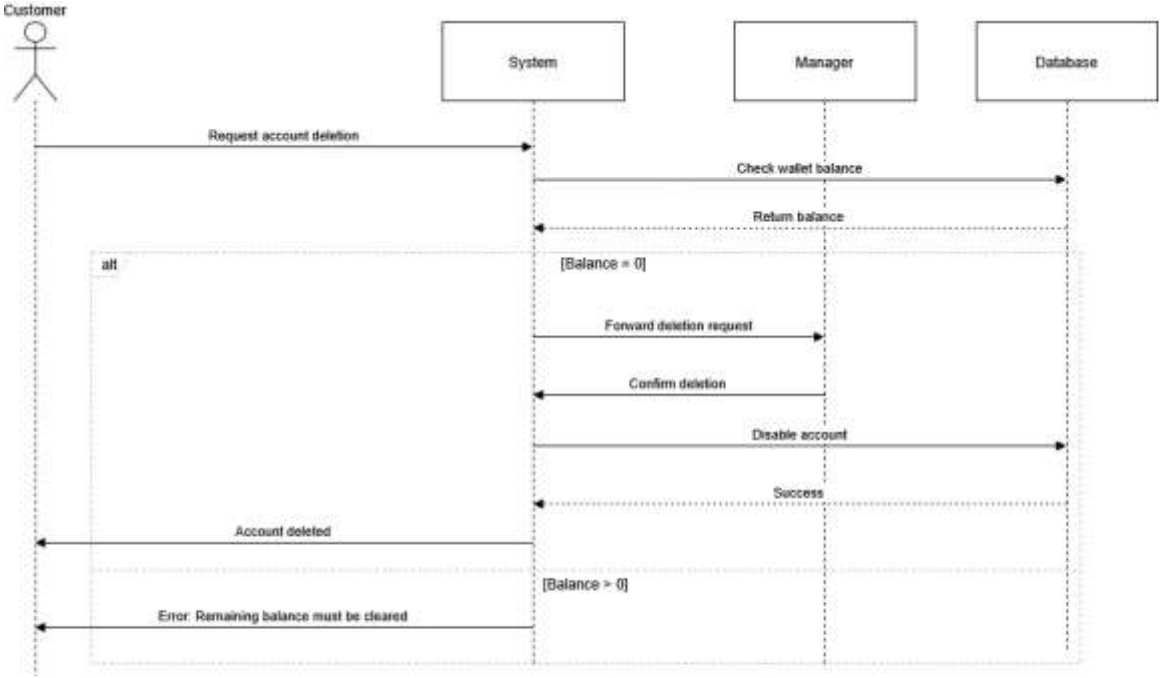
Main Flow:

1. Check if the account has remaining deposit
2. If conditions met, deregistration request delegates to manager to handle
3. Manager clears the account deposit
4. Manager close the account

Exceptional Scenario:

- Wallet balance > 0 → System rejects the deletion request.
- Manager rejects deletion → Customer notified.
- System auto-deletes account after the 3rd warning → no Manager approval required.

Database update failure → System cannot disable the account.



UC-007: Browse Menu

Description: Users browse restaurant menu

Actors: All user types

Main Flow:

1. User accesses homepage or menu page
2. System displays all available dishes
3. User can view dish details, prices, ratings
4. System shows personalized recommendations based on user type

Exceptional Scenario:

- No dishes available → System displays “No dishes available.”
- Database query failure → System shows fallback UI / error message.

UC-008: View Dish Details

Description: User views detailed information of specific dish

Actors: User, System

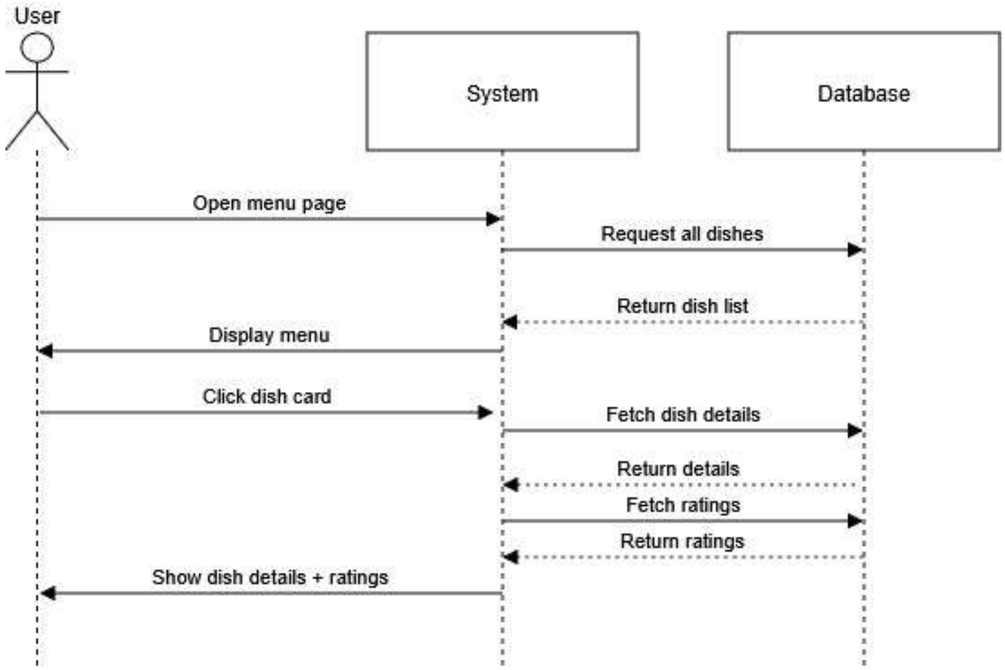
Main Flow:

1. User clicks on dish card
2. System displays detailed dish information
3. Shows user reviews and ratings
4. Provides option to add to cart

Exceptional Scenario:

- Dish not found (deleted/disabled) → System displays “Dish unavailable.”
- Rating fetch fails → System displays dish data without ratings.

- Database error → System shows error message or fallback UI.



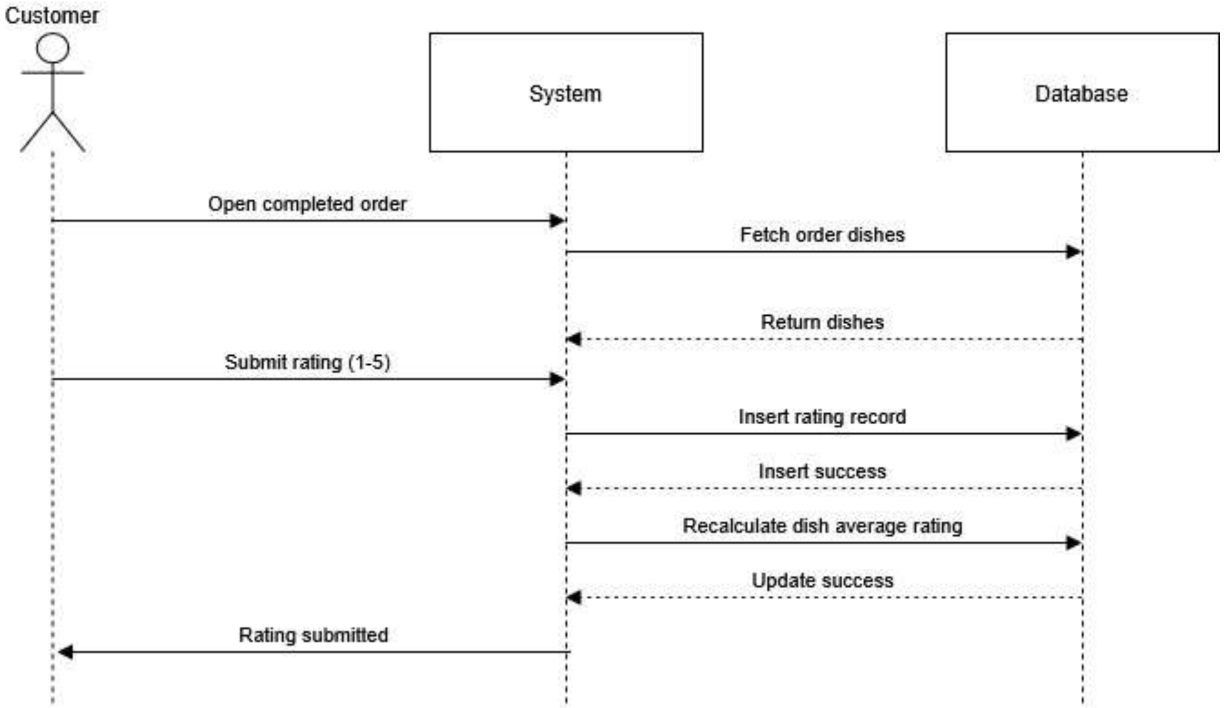
UC-009: Rate Dish

Description: Customer rates dish(es) in the purchased order

Actors: Customer, System

Main Flow:

1. User clicks one purchased order
2. System displays dish(es) with rating options
3. Customer rates each dish from 1 to 5 stars



UC-010: Create Dish

Description: Chef creates new dish

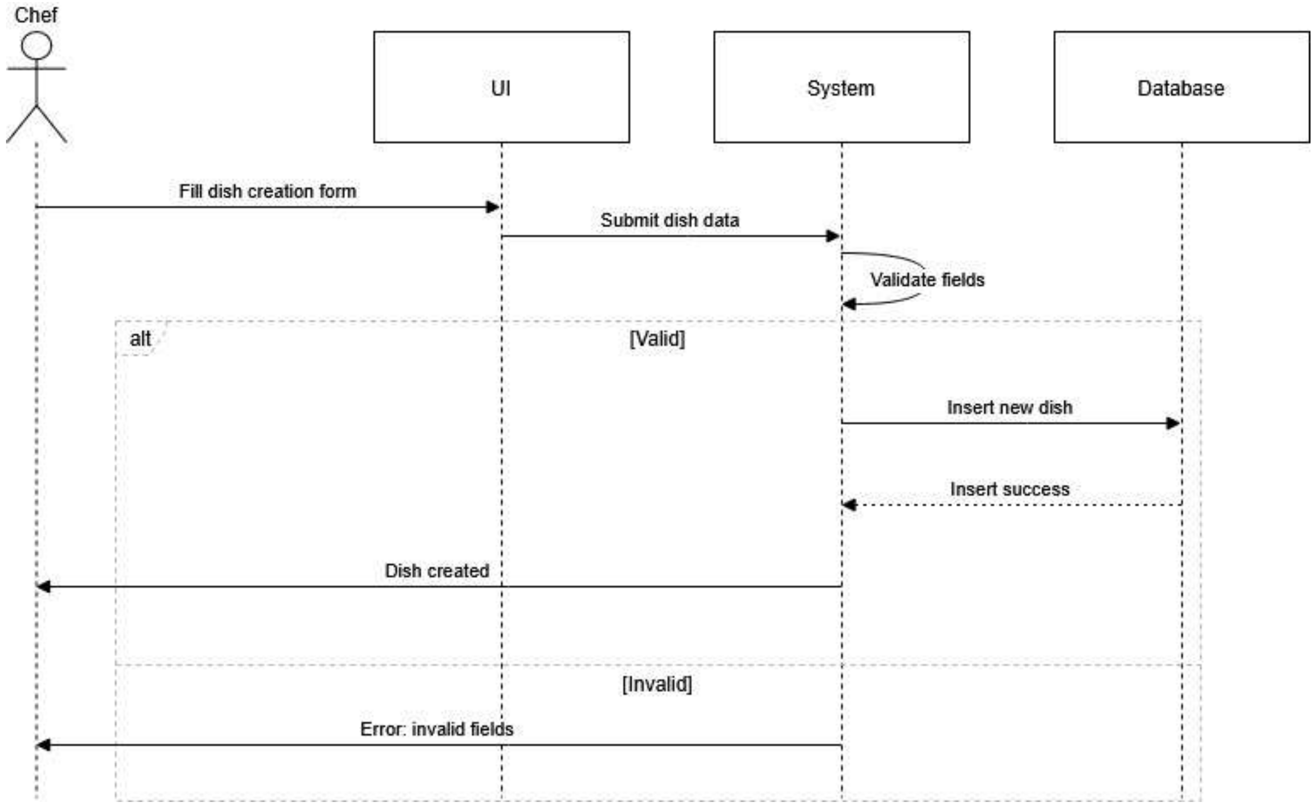
Actors: Chef, System

Main Flow:

1. Chef accesses dish management page
2. Fills in dish information (name, description, price, image)
3. Sets whether VIP exclusive
4. System saves dish information

Exceptional Scenario:

- Required fields are missing.
- Price is invalid (0 or negative).
- Database insertion error.



UC-011: Add Item to Cart

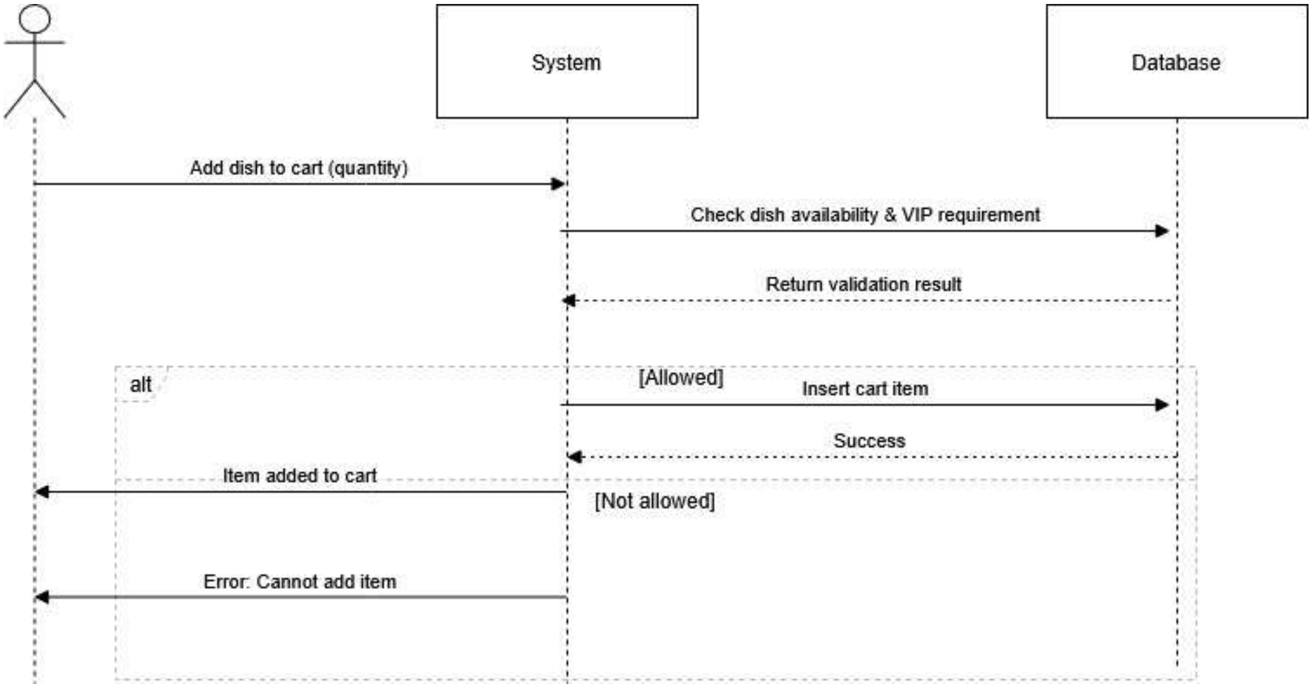
Description: Customer adds dish to shopping cart

Actors: Customer, System

Main Flow:

1. Customer selects dish and quantity
2. System checks VIP permissions (if applicable)
3. Adds to shopping cart
4. Updates cart display

Exceptional Scenario: database insertion fails



UC-012: Checkout & Payment Verification

Description: Verify order payment capability

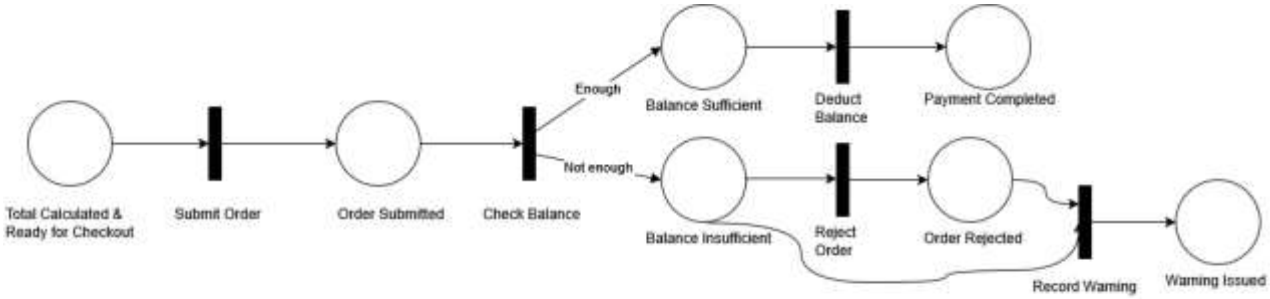
Actors: System

Main Flow:

1. Customer submits the order
2. Calculate order total amount
3. Check account balance
4. If insufficient balance, reject order and customer receives 1 warning
5. If sufficient balance, deduct fees

Exceptional Scenario:

Balance insufficient →System rejects order →System creates a warning record for the user.



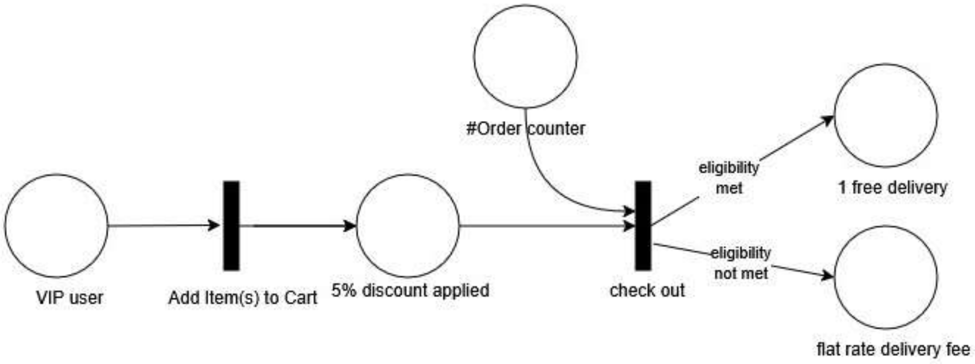
UC-013: VIP Privilege Application

Description: Apply privileges for VIP customers

Actors: System, VIP Customer

Main Flow:

1. Apply 5% discount
2. Provide VIP exclusive dish access
3. Provide 1 free delivery for every 3 orders



UC-014: Order Status Management

Description: System manages order status flow

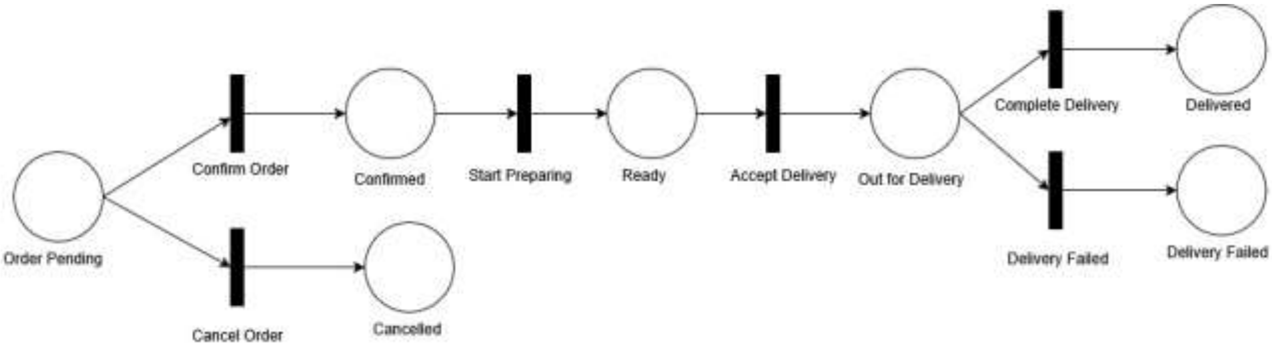
Actors: System, Chef, Delivery Person

Main Flow:

1. Order creation (pending)
2. Chef confirmation (confirmed)
3. Start preparation (preparing)
4. Preparation complete (ready)
5. Delivery person accepts (out_for_delivery)
6. Delivery complete (delivered)

Exceptional Scenario:

- Customer cancels while order is still pending. (Cancelled)
- Chef rejects or cannot prepare the order.
- Delivery fails → order marked as Failed.
- Database update error prevents status transition.



UC-015: Delivery Bidding

Description: Delivery person bids on orders

Actors: Delivery Person, System

Main Flow:

1. Delivery person views available orders
2. Enters bid price
3. System records bid information

Exceptional Scenario:

- Bid price is invalid (zero or negative).
- Order already assigned → System blocks bidding.
- Database insertion failure.

UC-016: Assign Bid

Description: Manager assigns bid on delivery person

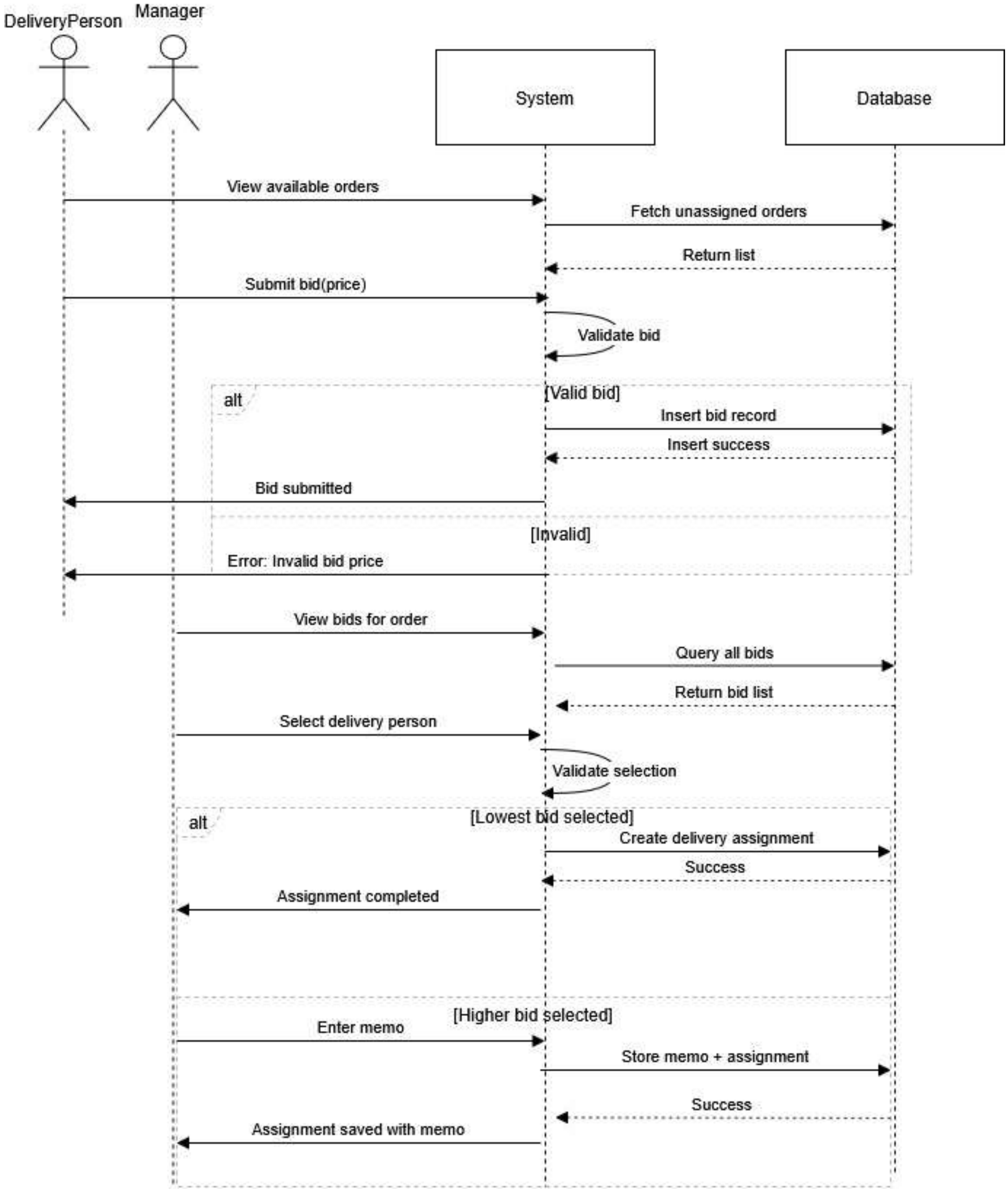
Actors: Manager, System

Main Flow:

1. Manager receives bid information from more than 1 delivery person
2. Manager reviews and selects delivery person offers the lowest price
3. If manager doesn’t select the lowest price, leave a memo in the system

Exceptional Scenario:

- Manager selects a higher bid → System prompts for a memo justification.
- No bids available → System blocks assignment.
- Database update failure for assignment or memo.


DeliveryPerson
Manager
System
Database
View available orders
Fetch unassigned orders
Return list
Submit bid(price)
Validate bid
alt
[Valid bid]
Insert bid record
Insert success
Bid submitted
[Invalid]
Error: Invalid bid price
View bids for order
Query all bids
Return bid list
Select delivery person
Validate selection
alt
[Lowest bid selected]
Create delivery assignment
Success
Assignment completed
[Higher bid selected]
Enter memo
Store memo + assignment
Success
Assignment saved with memo

UC-017: Delivery Route Planning

Description: System plans optimal route for delivery person

Actors: System, Delivery Person

Main Flow:

1. Obtains restaurant and customer address coordinates
2. Calls OpenRouteService API
3. Calculates optimal route
4. Displays route map and navigation information

Exceptional Scenario:

- ORS API fails or times out → System shows error and fallback instructions.
- Coordinates missing or invalid → System shows “Invalid address.”
- Network failure

UC-018: Update Delivery Status

Description: System update the distance from delivery person to the customer address

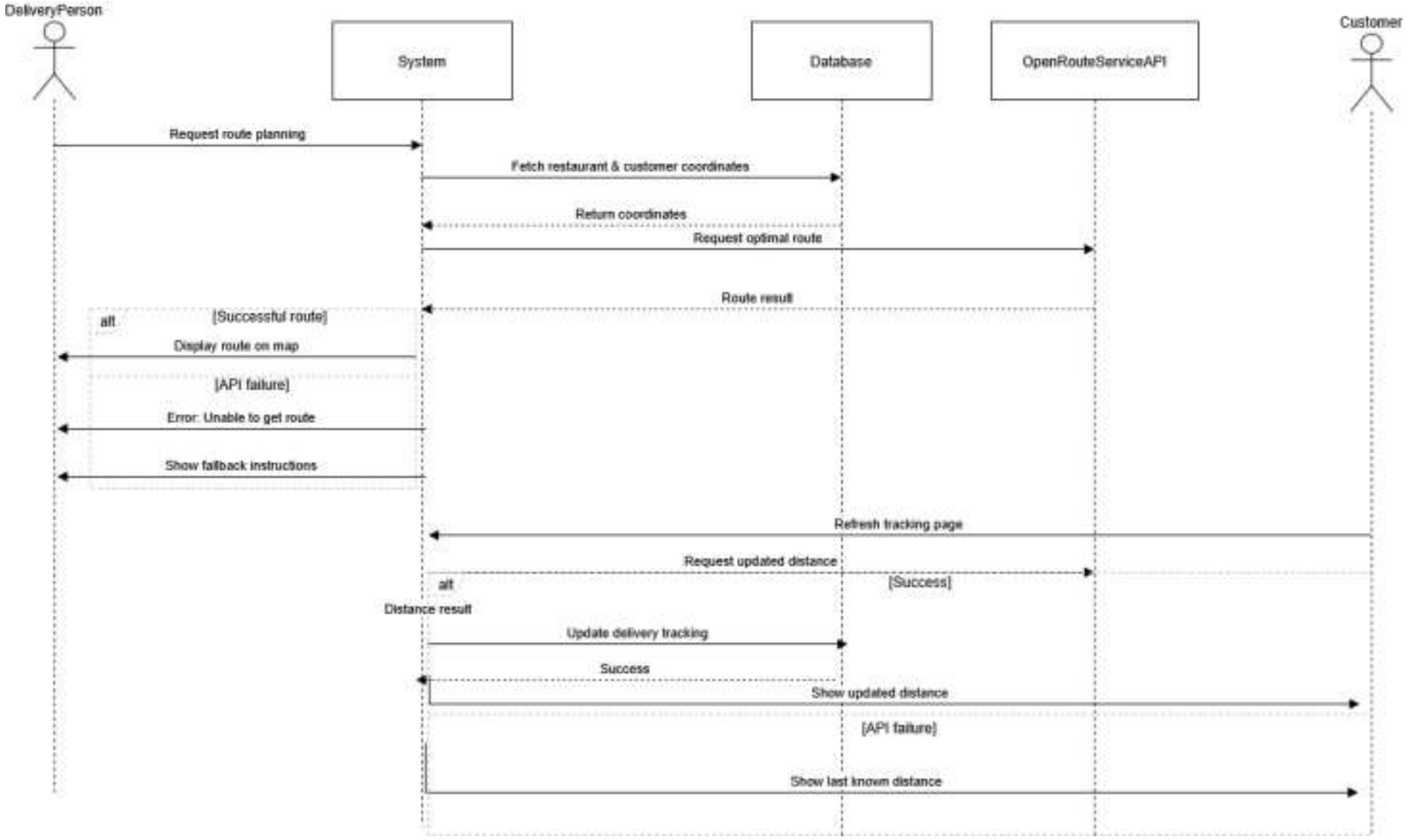
Actors: Customer, System, Delivery Person

Main Flow:

1. Every 5 minute or customer refreshes delivery tracking page
2. Calls OpenRouteService API
3. Updates the current distance from delivery person to the customer address

Exceptional Scenario:

- API error/timeout → System shows last known distance.
- Customer refreshes too quickly → System applies rate-limiting.
- GPS/location unavailable → System shows “Tracking temporarily unavailable.”



UC-019: Inquiry

Description: Users obtain restaurant information through local knowledge base and AI assistant

Actors: User, AI System

Main Flow:

1. User inputs question
2. System searches local knowledge base
3. If matching answer found, returns result
4. Otherwise calls Ollama LLM to generate answer

Exceptional Scenario:

- Knowledge base search fails → System falls back to LLM.
- LLM API not available → System shows “Unable to answer now.”
- User submits empty question → System rejects submission.

UC-020: Response Quality Rating

Description: User rates the quality of the response from local knowledge base

Actors: User, System

Main Flow:

1. System returns the response retrieves from the local knowledge base
2. Chat box displays rating options for the quality of content
3. User rates from 0 to 5 starts
4. If the rating is 0, the contents is labeled and sent to manager

Exceptional Scenario:

- User submits rating with no answer loaded → System blocks action.
- Rating value outside 0–5 → System rejects.
- Database insertion error → Rating not saved.

UC-021: Provide Content

Description: Customer posts content related to the restaurant

Actors: Customer, System

Main Flow:

1. Customer provides restaurant-related information
2. System stores in local knowledge base

Exceptional Scenario:

- Missing title or content ➝ System returns error and requests correction.
- Content too short or low quality ➝ System rejects submission.
- Database insertion failure ➝ Content not saved.

UC-022: Remove Content & Disenable user to post

Description: Manager handle 0 rating content

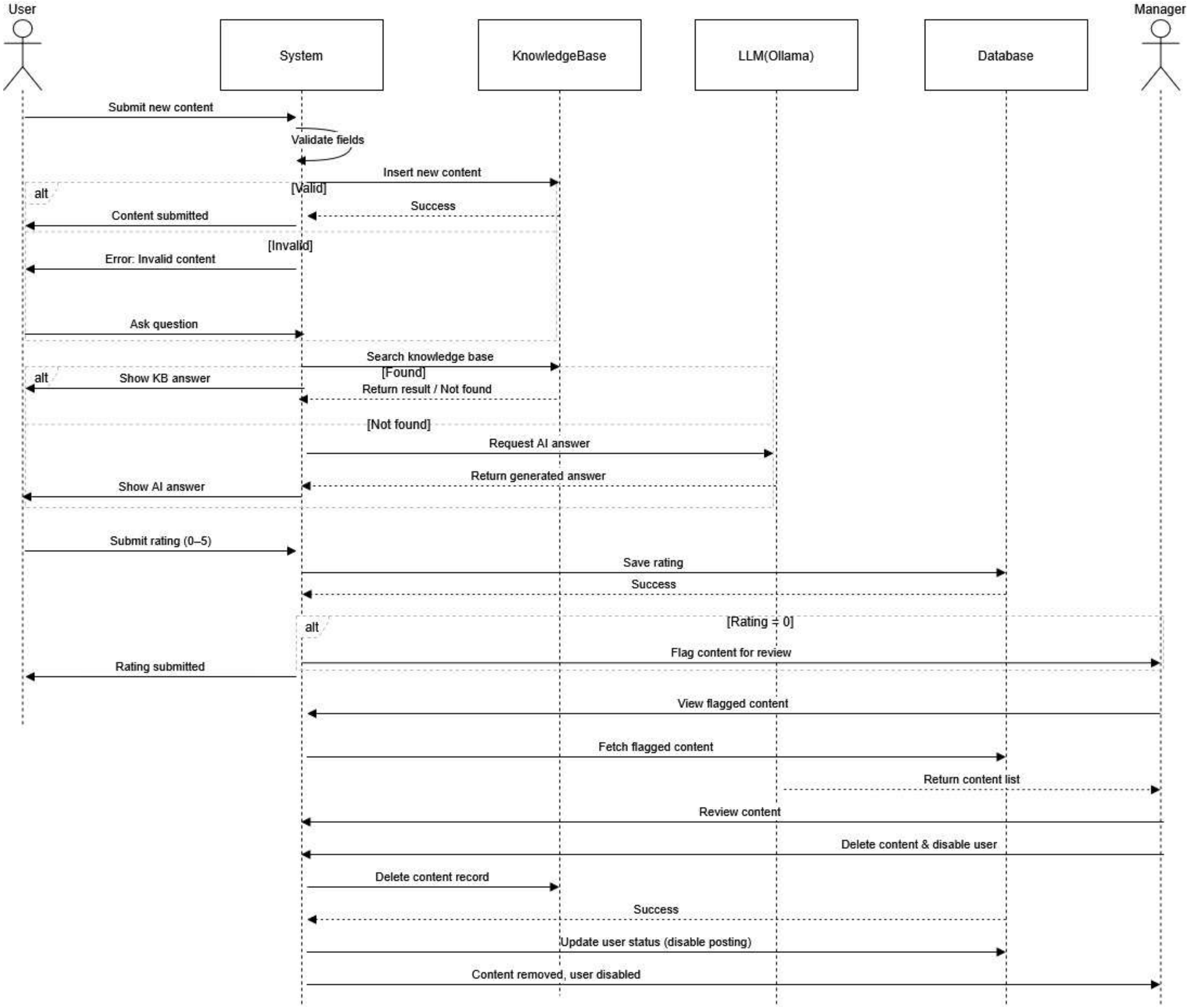
Actors: Manager, System

Main Flow:

1. Manager review the labeled content
2. If the content is bad indeed, delete the content from forum
3. Disenable the user to post content again

Exceptional Scenario:

- Content already removed ➝ System shows “Content unavailable.”
- Manager decides content is not harmful ➝ System unflags content.
- DB deletion or update failure ➝ Operation not completed.


User
System
KnowledgeBase
LLM(Ollama)
Database
Manager
Submit new content
Validate fields
Insert new content
alt
[Valid]
Success
Content submitted
[Invalid]
Error: Invalid content
Ask question
Search knowledge base
[Found]
alt
Show KB answer
Return result / Not found
[Not found]
Request AI answer
Return generated answer
Show AI answer
Submit rating (0–5)
Save rating
Success
alt
[Rating = 0]
Flag content for review
Rating submitted
View flagged content
Fetch flagged content
Return content list
Review content
Delete content & disable user
Delete content record
Success
Update user status (disable posting)
Content removed, user disabled

UC-023: Submit Complaint / Compliment

Description: User submits complaint / compliment

Actors: Customer, System, Manager

Main Flow:

1. User selects complaint / compliment target (chef/delivery person/customer)
2. Fills in complaint / compliment details
3. System records complaint / compliment information
4. Manager reviews and handles / confirms

Exceptional Scenario:

- Missing target or message → System rejects submission.
- Invalid complaint type → System returns an error.
- Database insertion failure → Complaint not saved.

UC-024: Dispute Complaint

Description: User dispute complaint against him/her

Actors: User, System, Manager

Main Flow:

1. System sends a notification to the person who receives complaint

3. System provides option for user to submit a dispute

3. If the user chooses to dispute, message is required to fill in

4. Dispute message is sent to manager to reviews

Exceptional Scenario:

- Dispute message empty → System requests proper message.
- Database insertion error → Dispute not saved.

UC-025: Complaint Handling

Description: Manager handles user complaints

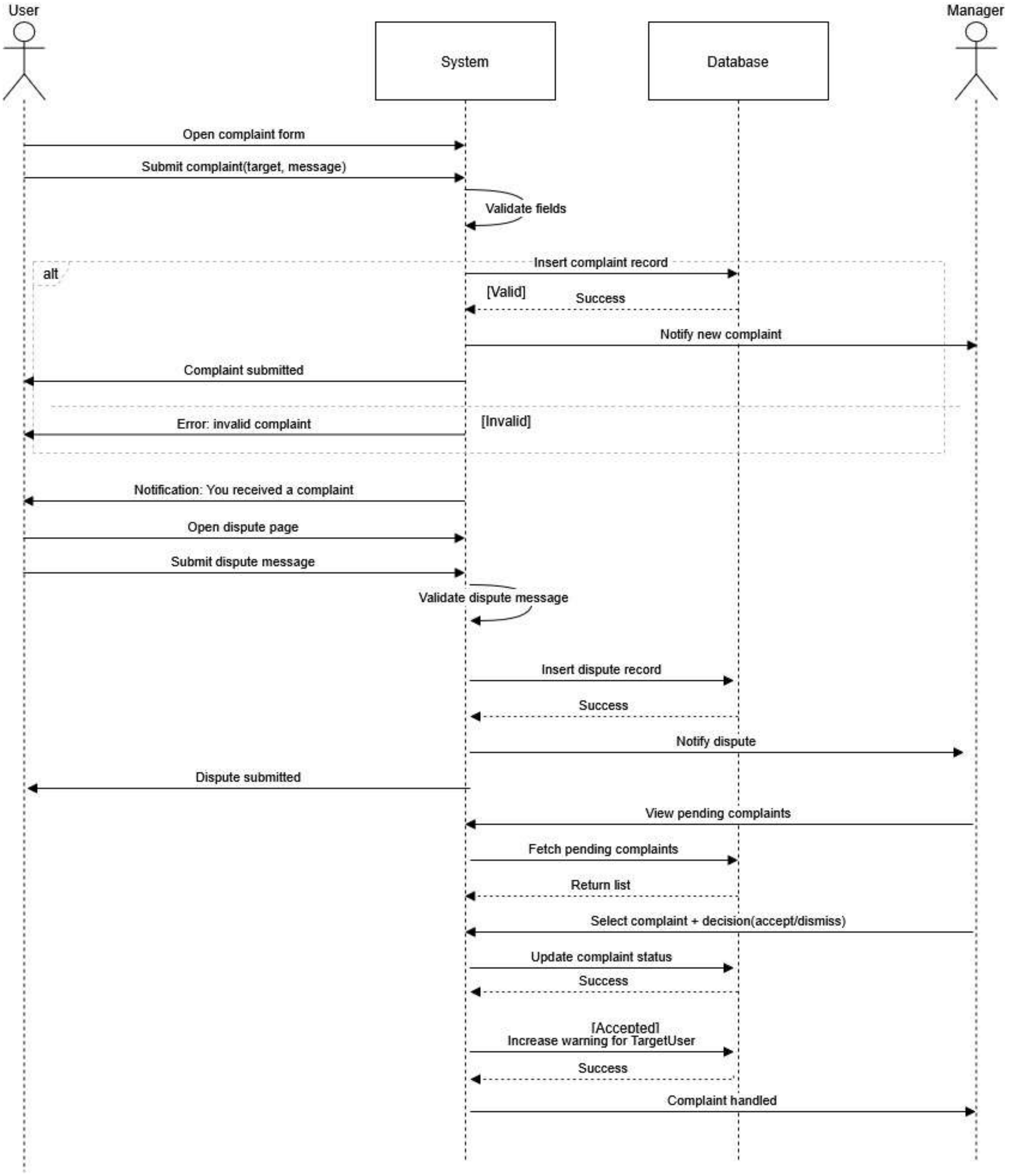
Actors: Manager, System

Main Flow:

1. Manager views pending complaints
2. Investigates complaint content
3. Makes handling decision (accept/dismiss)
4. Updates related user warning status

Exceptional Scenario:

- Manager does not select a decision → System requests action.
- Database update failure → Status or warning not updated.


User
System
Database
Manager
Open complaint form
Submit complaint(target, message)
Validate fields
alt
Insert complaint record
[Valid]
Success
Notify new complaint
Complaint submitted
Error: invalid complaint
[Invalid]
Notification: You received a complaint
Open dispute page
Submit dispute message
Validate dispute message
Insert dispute record
Success
Notify dispute
Dispute submitted
View pending complaints
Fetch pending complaints
Return list
Select complaint + decision(accept/dismiss)
Update complaint status
Success
[Accepted]
Increase warning for TargetUser
Success
Complaint handled

UC-026: Promote Staff

Description: Manager promote staff based on rating and compliment record

Actors: Manager, System

Main Flow:

1. Manager views latest order rating and compliment record
2. Confirm the authenticity of the content
3. Updates the evaluation (rating and number of compliments / complaint) of the staff
4. If rating > 4 stars or reaches the 3th compliment, offer staff bonus

Exceptional Scenario: Staff does not meet promotion criteria → System rejects promotion.

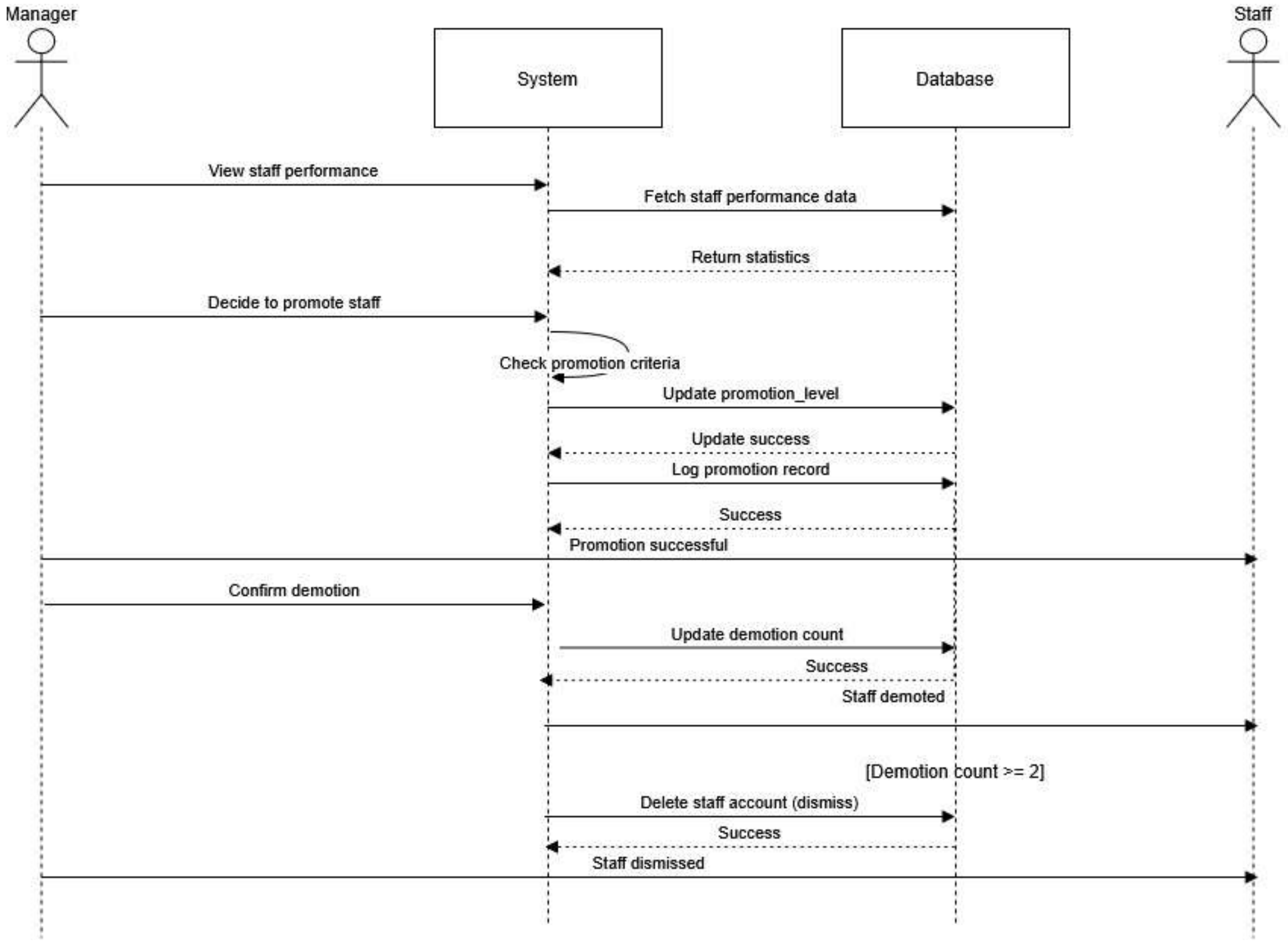
UC-027: Demote / Dismiss Staff

Description: Manager demote / dismiss staff based on rating and complaint record

Actors: Manager, System

Main Flow:

1. Manager views latest order rating and complaint record
2. Confirm the authenticity of the content
3. Updates the evaluation (rating and number of compliments / complaint) of the staff
4. If consistently rating < 2 or reaches the 3th complaints, the staff is demoted
5. If times of demotion reaches 2, the staff is dismissed by manager, related account is deleted



3.2 Supplementary Requirements

FR-001: User Interface Requirements

- System shall provide responsive web interface supporting desktop and mobile devices
- Interface shall use Bootstrap 5 framework ensuring modern design
- All pages shall include navigation bar, footer, and AI chat component

FR-002: Data Management Requirements

- System shall use Django ORM for data operations
- Support SQLite database storage
- Implement data backup and recovery functionality
- Support data import and export functionality

FR-003: Security Requirements

- Implement user authentication and authorization mechanisms
- Use Django built-in security features
- Support CSRF protection
- Implement session management
- Encrypt password storage

FR-004: Integration Requirements

- Integrate local Ollama LLM service for AI Q&A
- Integrate OpenRouteService API for route planning
- *Support RESTful API interfaces*
- Support JSON data exchange

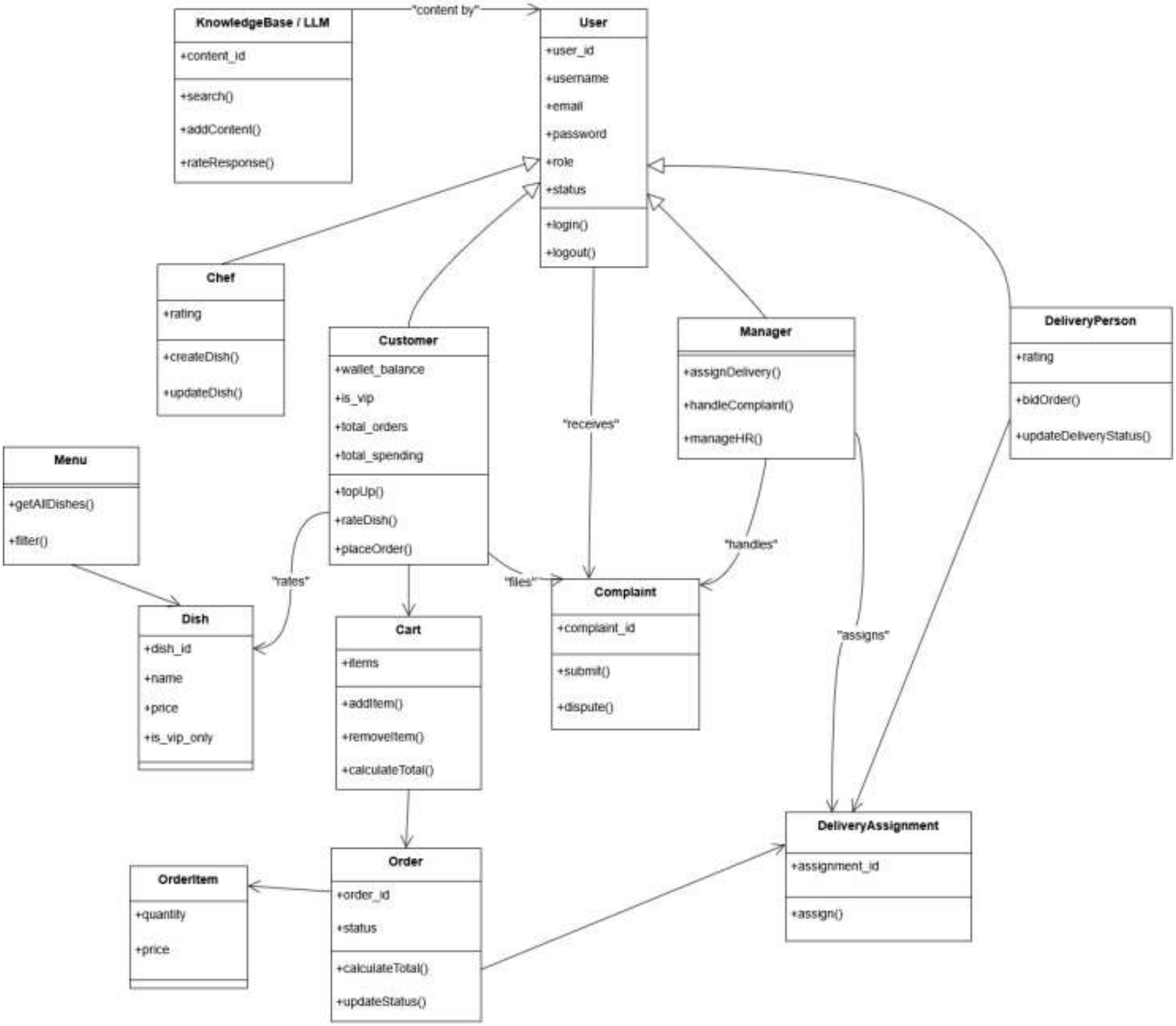
4. Supporting Information

4.1 Index

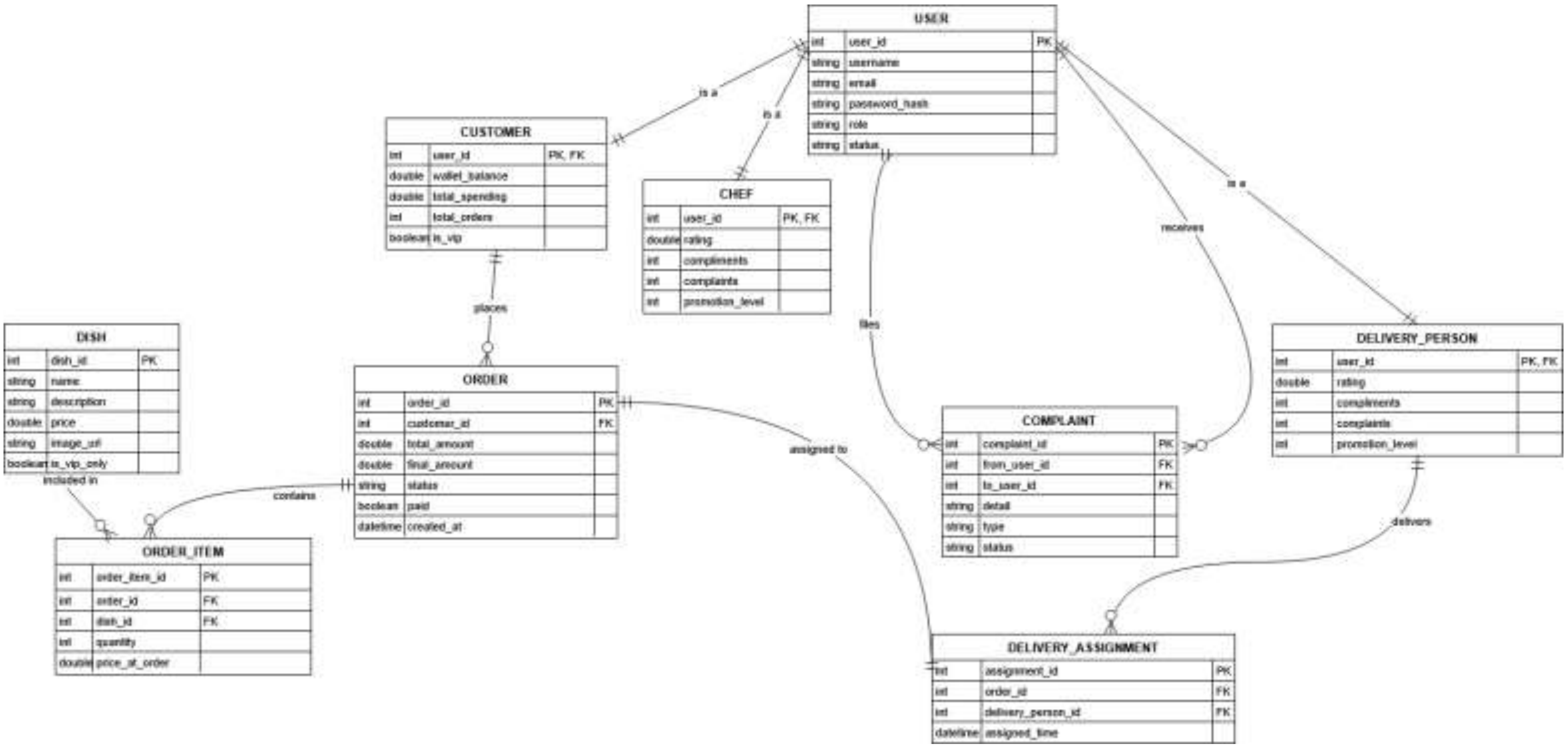
- Account Management Sub-System: UC-001, UC-002, UC-003, UC-004, UC-005, UC-006
- Menu and Dish Management Sub-System: UC-007, UC-008, UC-009, UC-010
- Order processing and Payment Sub-System: UC-011, UC-012, UC-013, UC-014
- Delivery Management Sub-System: UC-015, UC-016, UC-017, UC-018
- Knowledge Base & AI Customer Service Sub-System: UC-019, UC-020, UC-021, UC-022,
- Reputation and Human Resources Management Sub-System: UC-023, UC-024, UC-025, UC-026, UC-027

5. System Collaboration Diagram

The following collaboration diagram is a simplified version.



6. System E/R Diagram



7. System Pseudo-Code

Note: For our ongoing developing system with 27 use-cases in total, we think that attaching (may more than) 27 methods' pseudo-code here is too much. Therefore we only attached 3 crucial methods that have been implemented in the systems as demonstrative examples.

7.1 check_vip_status

```
METHOD: check_vip_status
INPUT:
    - self: Customer object instance

OUTPUT:
    - Boolean (true if customer is VIP, false otherwise)
    - Side effect: Updates customer.is_vip field if conditions met

MAIN FUNCTIONALITY:
BEGIN
    // Check VIP eligibility criteria
    IF customer.total_spent >= 100 OR customer.order_count >= 3 THEN
        // Check for unresolved complaints
        COUNT unresolved_complaints WHERE:
            complainant = customer AND
            status IN ['pending', 'investigating']

        IF unresolved_complaints == 0 THEN
            SET customer.is_vip = true
            SAVE customer
            RETURN true
        END IF
    END IF
    RETURN customer.is_vip
END
```

7.2 ai_chat

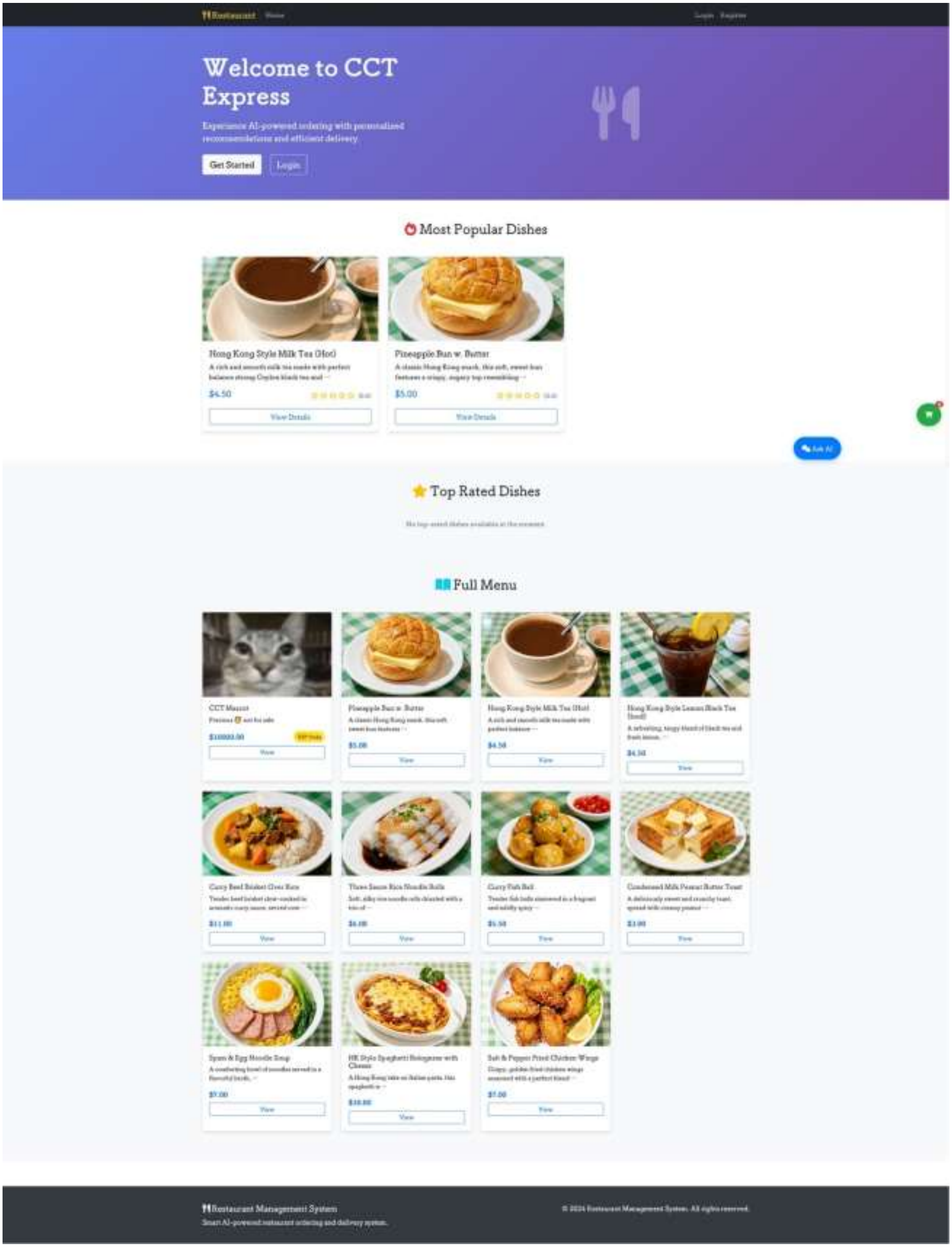
```
INPUT:
    - request: HTTP request object containing JSON body
    - question: String (customer's question about restaurant/food)

OUTPUT:
    - JsonResponse containing:
        * answer: String (response text)
        * source: String ('local', 'LLM', or 'fallback')
        * knowledge_id: Integer (optional, if from local knowledge base)
        * model: String (optional, LLM model name if used)
    OR error response with status code

MAIN FUNCTIONALITY:
BEGIN
    // Parse input
    PARSE JSON data from request body
    EXTRACT question string
    TRIM whitespace from question

    IF question is empty THEN
        RETURN error response (400 Bad Request)
    END IF

    // Step 1: Search local knowledge base
    SEARCH KnowledgeBase WHERE:
        (question CONTAINS question OR answer CONTAINS question) AND
        is_flagged = false
    ORDER BY created_at DESCENDING
    LIMIT to 3 results

    IF knowledge entries found THEN
        GET first entry as best_answer
        RETURN JsonResponse WITH:
            answer = best_answer.answer
            source = 'local'
            knowledge_id = best_answer.id
    END IF

    // Step 2: Delegate to LLM (Ollama)
    TRY
        CONSTRUCT LLM prompt = "You are a helpful restaurant assistant.
                                Answer this customer question about our
restaurant: {question}.
                                Be friendly and helpful."

        SEND POST request to 'http://localhost:11434/api/generate' WITH:
            model = 'llama3:8b'
            prompt = constructed prompt
            stream = false
        SET timeout = 30 seconds

        IF response status code == 200 THEN
            PARSE JSON response
            EXTRACT answer from response
            RETURN JsonResponse WITH:
                answer = extracted answer
                source = 'LLM'
                model = 'llama3:8b'
        ELSE
            RETURN fallback response
        END IF

    CATCH RequestException THEN
        RETURN fallback response WITH:
            answer = "I'm sorry, I don't have specific information about that
in our knowledge base,
                     and our AI assistant is currently unavailable.
                     Please contact our staff for assistance."
            source = 'fallback'
    END TRY

    RETURN fallback response
END
```

7.3 handle_employee_demotion

```
INPUT:
    - sender: Model class (Complaint)
    - instance: Complaint object instance
    - created: Boolean (true if new record created)
    - **kwargs: Additional keyword arguments

OUTPUT:
    - None (void function, side effects only)
    - Side effects: Updates employee status, demotion_count, and is_active flag

MAIN FUNCTIONALITY:
BEGIN
    // Only process newly created complaints that are resolved
    IF NOT created OR instance.status != 'resolved' THEN
        RETURN (exit early)
    END IF

    // Handle chef demotions
    IF instance.chef is not null THEN
        GET chef = instance.chef
        INCREMENT chef.demotion_count by 1

        IF chef.demotion_count >= 3 THEN
            // Demote chef (reduce salary)
            APPLY salary reduction logic
            SAVE chef
        ELSE IF chef.demotion_count >= 2 THEN
            // Fire chef after second demotion
            SET chef.is_active = false
            SET chef.is_terminated = true
            SAVE chef
        ELSE
            SAVE chef (update demotion_count only)
        END IF
    END IF

    // Handle delivery person demotions
    IF instance.delivery_person is not null THEN
        GET delivery_person = instance.delivery_person
        INCREMENT delivery_person.demotion_count by 1
        IF delivery_person.demotion_count >= 3 THEN
            // Demote delivery person (reduce salary)
            APPLY salary reduction logic
            SAVE delivery_person
        ELSE IF delivery_person.demotion_count >= 2 THEN
            // Fire delivery person after second demotion
            SET delivery_person.is_active = false
            SET delivery_person.is_terminated = true
            SAVE delivery_person
        ELSE
            SAVE delivery_person (update demotion_count only)
        END IF
    END IF

    RETURN
END
```

8. Mockups & Demo

- Homepage of the CCT Express, viewed as a visitor.



Restaurant Home

Login Register

Welcome to CCT Express

Experience AI-powered ordering with personalized recommendations and efficient delivery.

Get Started Login

Most Popular Dishes

Hong Kong Style Milk Tea (Hot)
A rich and smooth milk tea made with perfect balance strong Ceylon black tea and …
$4.50 ☆☆☆☆☆ [illegible]
View Details

Pineapple Bun w. Butter
A classic Hong Kong snack, this soft, sweet bun features a crispy, sugary top resembling …
$5.00 ☆☆☆☆☆ [illegible]
View Details

Ask AI

Top Rated Dishes

No top-rated dishes available at the moment.

Full Menu

CCT Mascot
Precious [illegible] not for sale
$10000.00 [illegible]
View

Pineapple Bun w. Butter
A classic Hong Kong snack, this soft, sweet bun features …
$5.00
View

Hong Kong Style Milk Tea (Hot)
A rich and smooth milk tea made with perfect balance …
$4.50
View

Hong Kong Style Lemon Black Tea (Iced)
A refreshing, tangy blend of black tea and fresh lemon. …
$4.50
View

Curry Beef Brisket Over Rice
Tender beef brisket slow-cooked in aromatic curry sauce, served over …
$11.00
View

Three Sauce Rice Noodle Rolls
Soft, silky rice noodle rolls drizzled with a trio of …
$6.00
View

Curry Fish Ball
Tender fish balls simmered in a fragrant and mildly spicy …
$5.50
View

Condensed Milk Peanut Butter Toast
A deliciously sweet and crunchy toast, spread with creamy peanut …
$3.00
View

Spam & Egg Noodle Soup
A comforting bowl of noodles served in a flavorful broth, …
$7.00
View

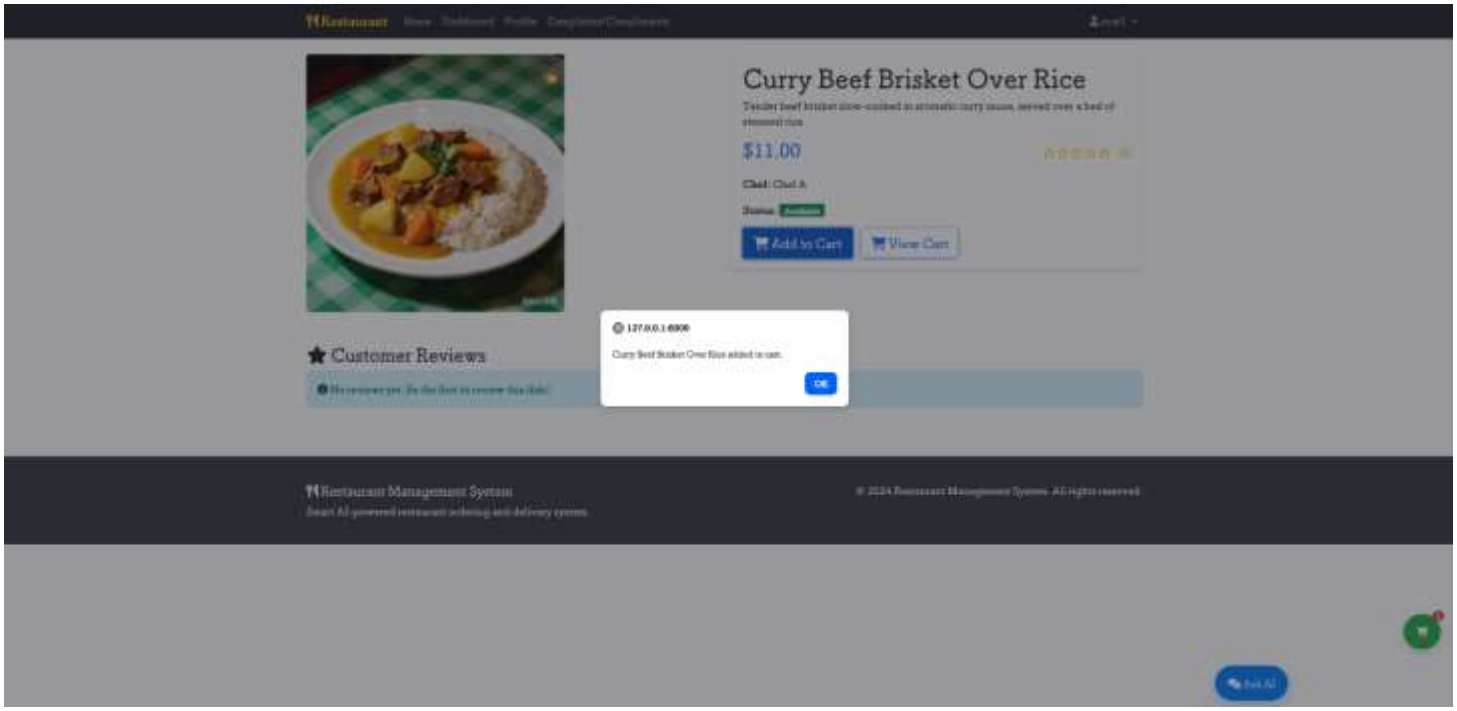
HK Style Spaghetti Bolognese with Cheese
A Hong Kong take on Italian pasta. This spaghetti is …
$10.00
View

Salt & Pepper Fried Chicken Wings
Crispy, golden fried chicken wings seasoned with a perfect blend …
$7.00
View

Restaurant Management System
Smart AI-powered restaurant ordering and delivery system.

© 2024 Restaurant Management System. All rights reserved.

- Regular customer add an item to the cart.



Restaurant
Home
Dashboard
Profile
Complaints/Compliments
Curry Beef Brisket Over Rice
Tender beef brisket slow-cooked in aromatic curry sauce, served over a bed of steamed rice.
$11.00
Chef: Chef A
Status: Available
Add to Cart
View Cart
Customer Reviews
No reviews yet. Be the first to review this dish!
Restaurant Management System
Smart AI-powered restaurant ordering and delivery system.
© 2024 Restaurant Management System. All rights reserved.
Ask AI

127.0.0.1:8000

Curry Beef Brisket Over Rice added to cart.

OK

9. Memos & Teamwork

As a two-person team, we quickly reached a consensus on several key decisions: using a GUI built as a web application, choosing our tech stack, and defining the conceptual model of the simulated company for the project. In the early stages, we spent a significant amount of time identifying 27 use cases for a system with multiple user roles and a wide range of complex functions (perhaps because we over-specified the requirements). However, when we attempted to map out the system logic, the process of creating diagrams still felt extremely chaotic. (Not all) team member began to question the practical value of diagrams in guiding a real-world project development. We are not trying to challenge their place in the CS education; on the contrary, creating various diagrams may indeed help cultivate structured thinking in system design.

What worries us more at the moment is whether it is even achievable for us to implement all the features of the system. This is our first time developing such a complex application, although we chose Django for its “quick and simple” benefits—ORM support, built-in templates, and an authentication/authorization system—the project’s complexity continues to pose significant challenges. We still have unresolved issues such as integrating an LLM with a local knowledge base and configuring OpenRouteService so it can successfully return data.

Since one team member mainly works on macOS, we considered containerizing the project with Docker at one point. But given that we need to connect to a local LLM model, Dockerization would introduce even more complications. As a result, both of us are now developing collaboratively in our respective Windows environments remotely.

10. Github Repository

https://github.com/yxhgs520/322_groupL_CCTExpress