322 Phase 2 Report

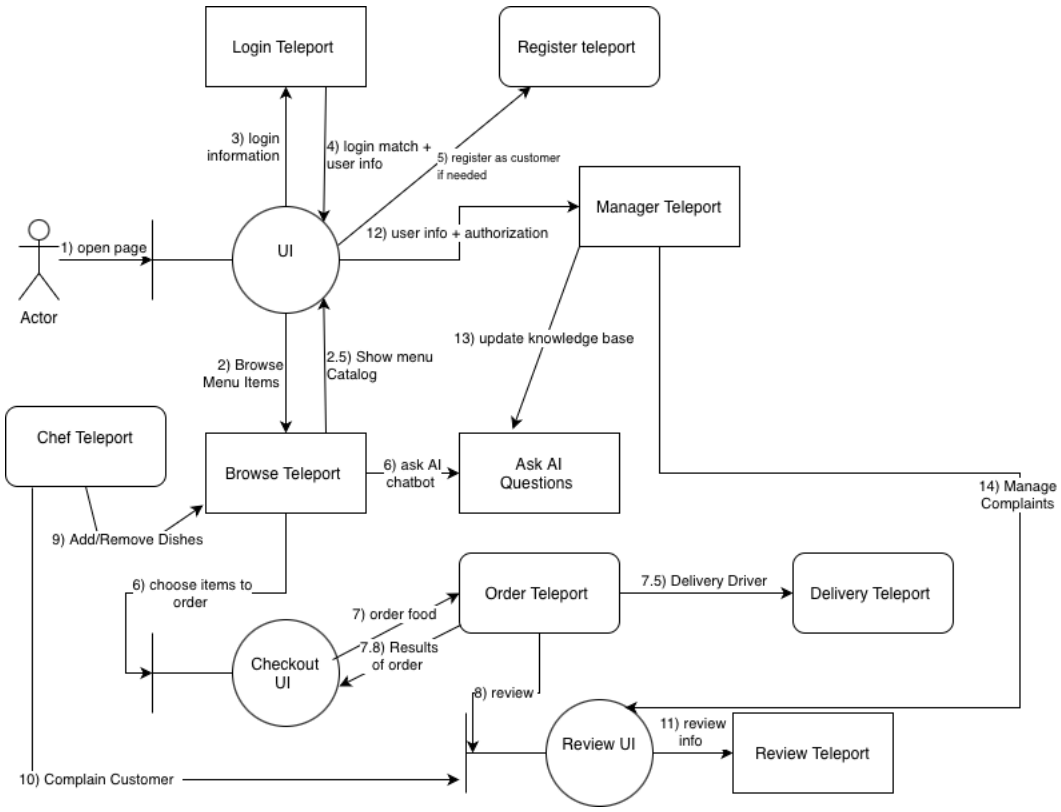
Group F -

Introduction

This report focuses on a high-level understanding and explanation of an AI enabled restaurant interface. The goal of the system is to integrate customer ordering, financial transactions, feedback processing, and AI-assisted interaction into a unified platform that supports multiple user roles. These roles include Visitors, Registered Customers, VIP Customers, Delivery Personnel, and Managers, each with distinct privileges and responsibilities. The system provides browsing and ordering experiences for customers, automates payment and deposit handling, and incorporates an AI assistant whose knowledge base is continuously improved.

To provide an overall understanding of how these users and components interact, a sequence diagram is included as a high-level overview. The diagram models the major system features such as Account, Menu, Order, Payment, and Review. While the diagram is intentionally not exhaustive, it captures the essential relationships among actors and system classes, illustrating how user actions propagate through controllers and data entities to achieve the required functionality. To see detailed reports visit the use case section of the report which uses class diagrams and petri nets to detail functionalities.

Section 3 defines the ER diagram of the system, specifying each entity's attributes and primary keys. Section 4 provides the detailed design of all methods through structured pseudo-code, outlining inputs, outputs, and core logic. Section 5 presents the primary system screens and includes a functional prototype of one selected feature.



Use Cases:

Login Petri Net:

Normal Scenario

1. Customer navigates to login screen.
2. Customer enters credentials.
3. System verifies username and password.
4. System loads personalized dashboard.
5. System displays account status and any warnings.

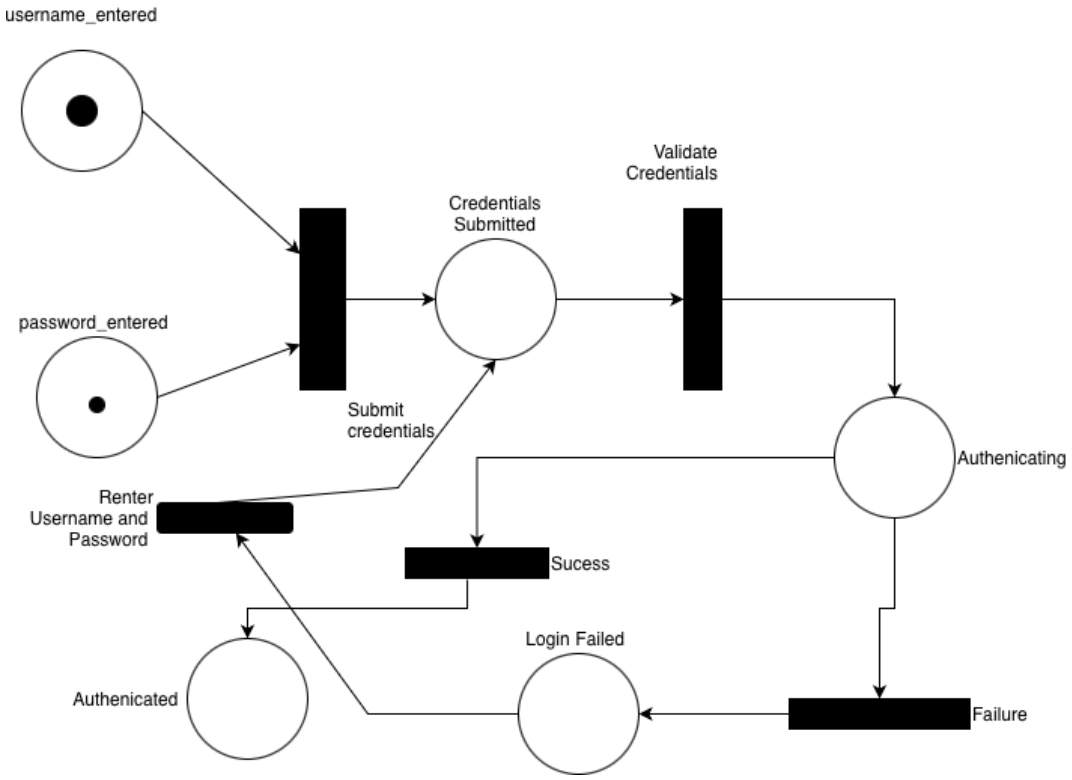
Exception Scenario A — Wrong Password

1. Customer enters incorrect password.
2. System rejects login.
3. System prompts retry or password recovery.

Exception Scenario B — Blacklisted or Closed Account

1. Customer attempts login.

2. System detects account is blacklisted or closed.
3. System denies access and shows status message.



Registration:

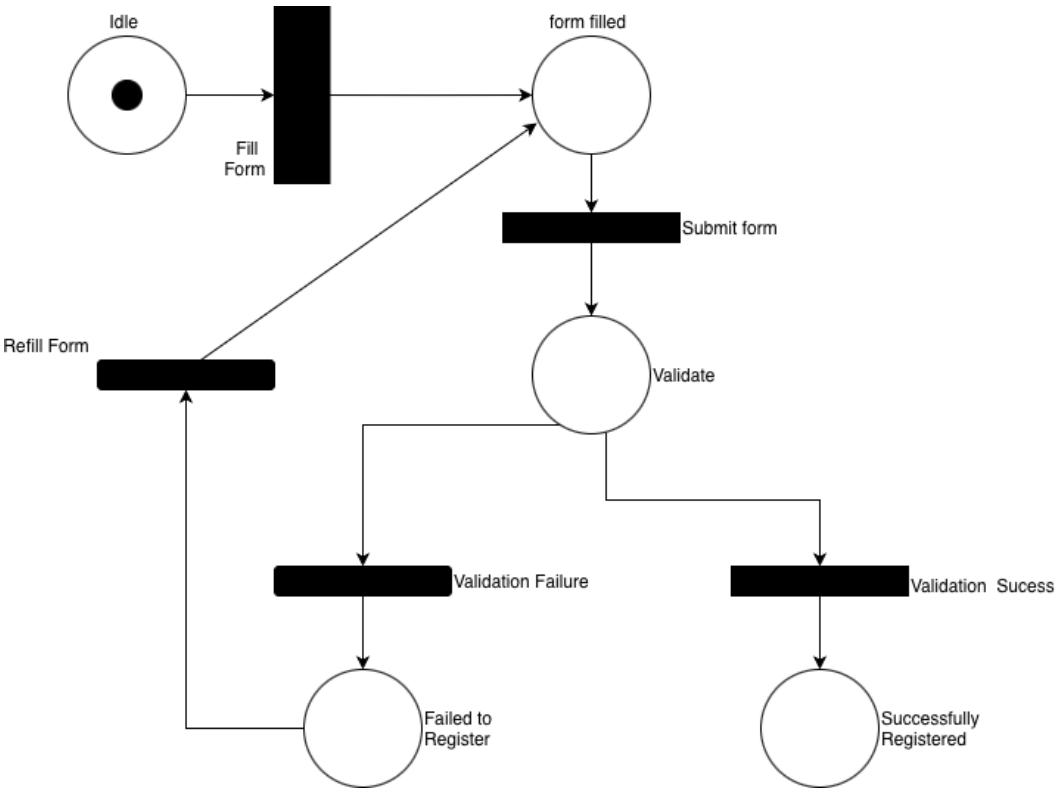
1) Visitor opens the registration page.
2) System displays the registration form.
3) Visitor enters required details (name, email, password, etc.).
4) System validates all fields.
5) System checks if the email is not already registered.
6) System creates a new customer account.
7) System confirms successful registration and prompts for login.

Exception Scenario A — Email Already Used

1) Visitor submits registration form.
2) System detects duplicate email.
3) System rejects registration and displays an error message.
4) Visitor is prompted to use another email or reset password.

Exception Scenario B — Missing or Invalid Information

1) Visitor submits incomplete or invalid data.
2) System identifies invalid fields.
3) System highlights errors and prevents account creation.
4) Visitor corrects data and resubmits.



Browsing:

Normal Scenario

1. Any user selects "Browse Menu."
2. System fetches menu items and descriptions.
3. For customers/VIPs, system retrieves order history.
4. System generates personalized menu ordering/prioritization.
5. User views all menu items, images, and details.

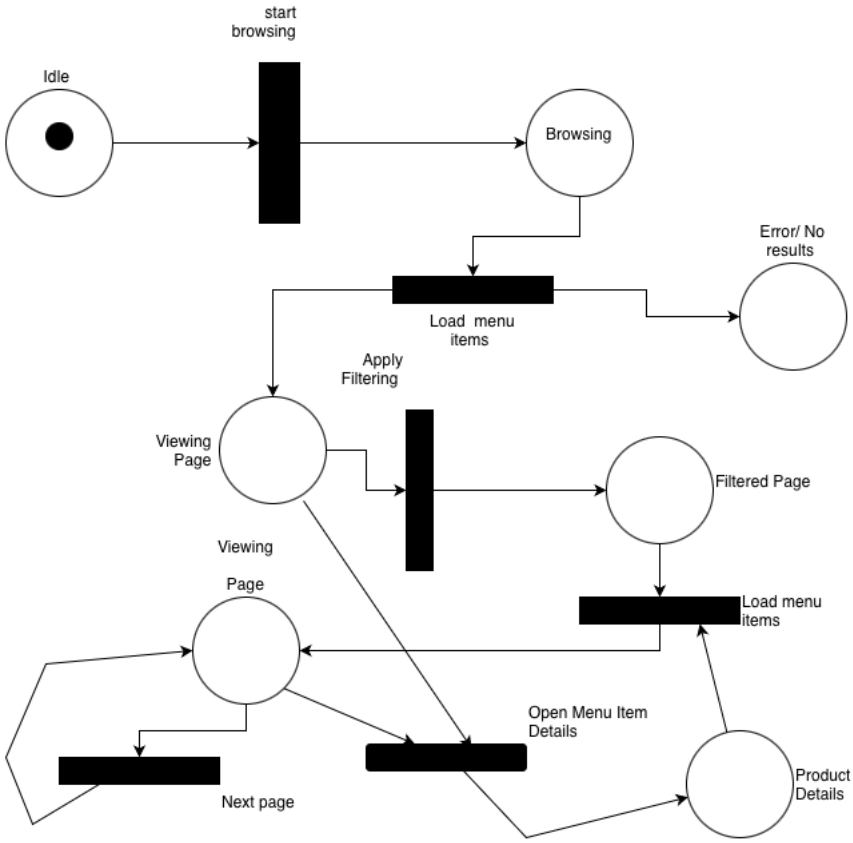
Exception Scenario A — Menu Items Unavailable

1. System attempts to load menu.

2. Database connection temporarily fails.
3. System notifies the user and displays cached menu if available.

Exception Scenario B — Personalization Data Missing

1. Customer logs in and browses menu.
2. System cannot retrieve order history.
3. System displays general menu without personalization.



Order Food:

Normal Scenario

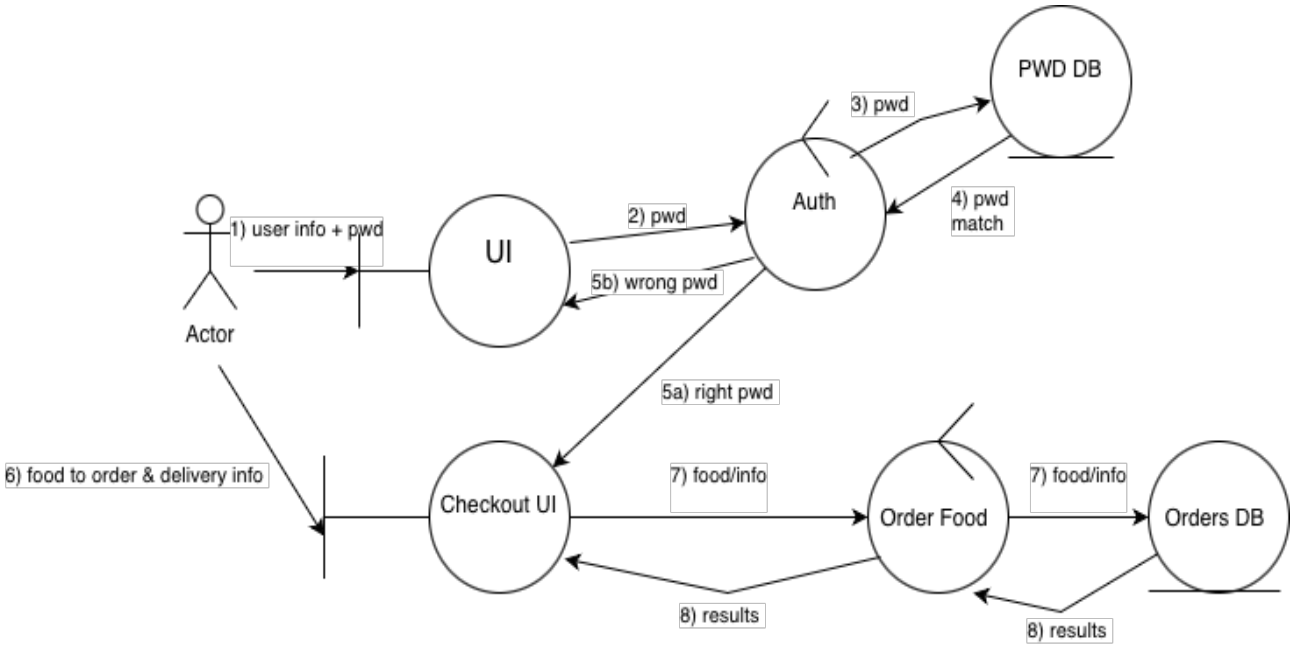
1. Registered or VIP customer selects items.
2. Customer adds items to cart.
3. Customer submits order.
4. System verifies sufficient deposit.
5. System processes order and stores order record.
6. System notifies delivery personnel of new order for bidding.

Exception Scenario A — Insufficient Deposit

1. Customer submits order.
2. System calculates total price.
3. System detects insufficient funds.
4. System blocks order and prompts to deposit more.

Exception Scenario B — Item Out of Stock

1. System checks stock.
2. One or more items are unavailable.
3. System rejects order and suggests alternatives.



Ask Questions:

Exceptional Scenario A — External LLM Unavailable

1. The user submits a question.
2. The system checks the local knowledge base but finds no matching entries.
3. The system attempts to query the external LLM.
4. The connection fails or the API returns an error.

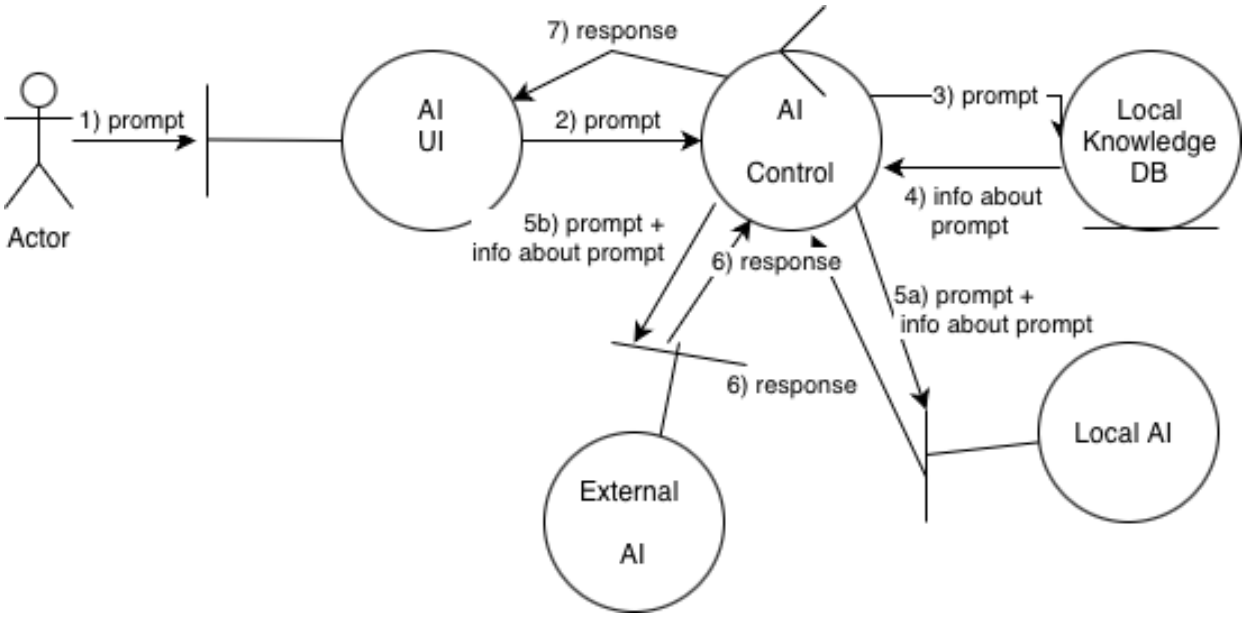
5. The system informs the user that external lookup is temporarily unavailable.
6. The system offers to retry, to search only local content, or to notify the user when service is restored.

Exceptional Scenario B — Response Fails Safety Check

1. The user submits a question.
2. The system retrieves an answer from the external LLM.
3. The safety filter detects harmful, offensive, or inappropriate content.
4. The system blocks the unsafe response.
5. The system provides a safe fallback message such as “I cannot answer that question, but I can help with related topics.”
6. The system flags the interaction and sends it to the manager’s review queue.
7. The user receives the safe fallback message instead of the unsafe one.

Exceptional Scenario C — User Submits Invalid or Empty Input

1. The user submits an empty message or invalid input (such as only spaces or unsupported characters).
2. The system detects invalid input.
3. The system prompts the user to enter a valid question.
4. The system waits for a corrected query before continuing.



Review Order:

Normal Scenario

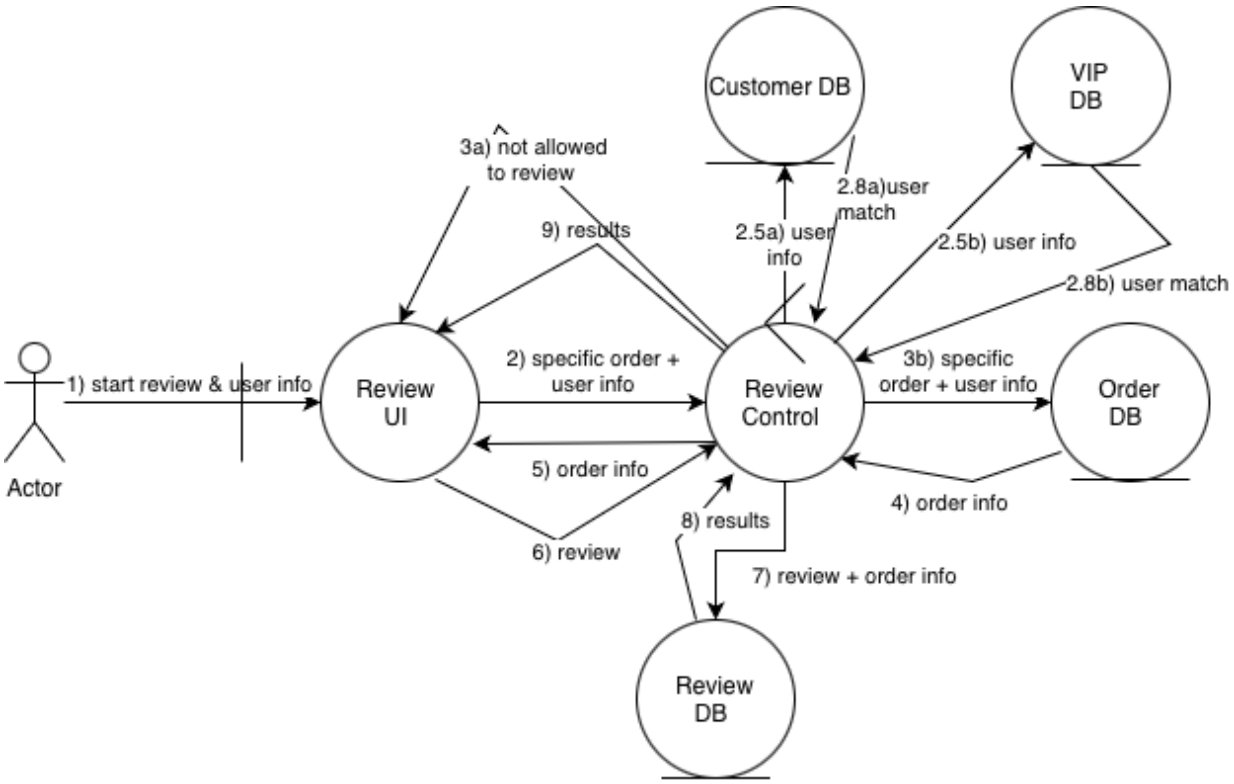
1. Customer receives order.
2. Customer selects "Rate Order."
3. System displays food and delivery rating form.
4. Customer submits ratings and comments.
5. System stores feedback and updates performance metrics.

Exception Scenario A — Customer Attempts to Rate Before Delivery

1. Customer tries to rate early.
2. System detects order still in progress.
3. System prevents rating and displays status.

Exception Scenario B — Missing Order Record

1. System attempts to fetch order.
2. Order ID invalid or missing.
3. System rejects rating submission.



Chef:

Remove/Add Dishes:

Normal Scenario

1. The chef accesses the “Menu Management” section of the system.
2. The system retrieves and displays the current menu, organized by categories.
3. The chef chooses whether to add a new dish or remove an existing one.
4. If adding a dish, the chef fills in required information such as dish name, description, price, category, image, and availability.
5. If removing a dish, the chef selects the dish from the menu listing.
6. The chef submits the add or remove request.
7. The system validates the provided information and checks for conflicts (such as duplicate dish names when adding).
8. The system updates the menu accordingly and stores the modification.
9. The system confirms the update and displays the revised menu to the chef.

Exceptional Scenario A Invalid Dish Data (Adding a Dish)

1. The chef attempts to add a dish with missing or invalid fields such as name, price, or category.
2. The system detects invalid or incomplete data.
3. The system highlights the problematic fields and prevents submission.
4. The chef corrects the data and resubmits the request.

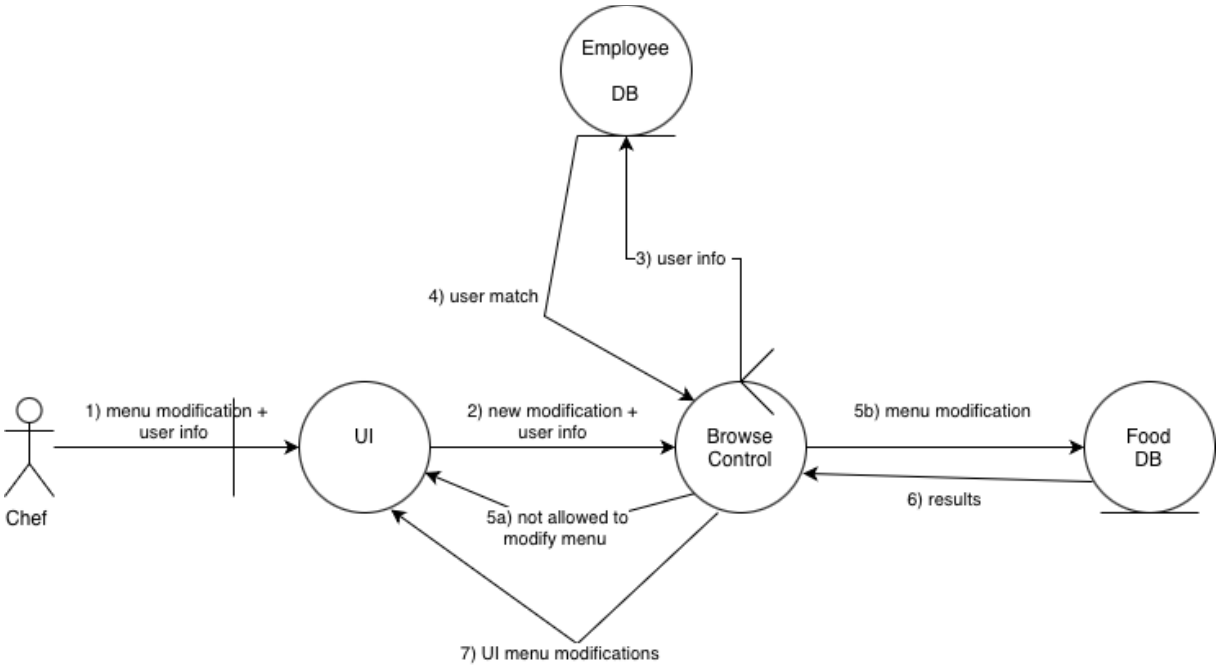
Exceptional Scenario B Dish Not Found (Removing a Dish)

1. The chef selects a dish to remove.
2. The system cannot locate the dish, possibly due to simultaneous edits or a recently deleted record.
3. The system notifies the chef that the dish could not be found.
4. The chef refreshes the menu and selects an available dish or cancels the action.

Exceptional Scenario C System Fails to Update Menu

1. The chef submits an add or remove request.
2. A database or system error prevents the menu from being updated.
3. The system notifies the chef that the update failed.

4. The chef may retry later after the system recovers.



Complain about customers:

Normal Scenario

1. Chef logs into the system and opens the “Customer Complaints” interface.
2. The system retrieves a list of customers associated with recent orders handled by that chef.
3. The chef selects the customer to complain about.
4. The system displays the complaint form, including predefined complaint categories and a section for optional comments.
5. The chef selects one or more complaint reasons and adds details if needed.
6. The chef submits the complaint.
7. The system validates the input and stores the complaint, linking it to the customer's record.

8. The system confirms the submission and notifies the manager's dashboard that a new complaint requires review.

Exceptional Scenario A Customer Not Found

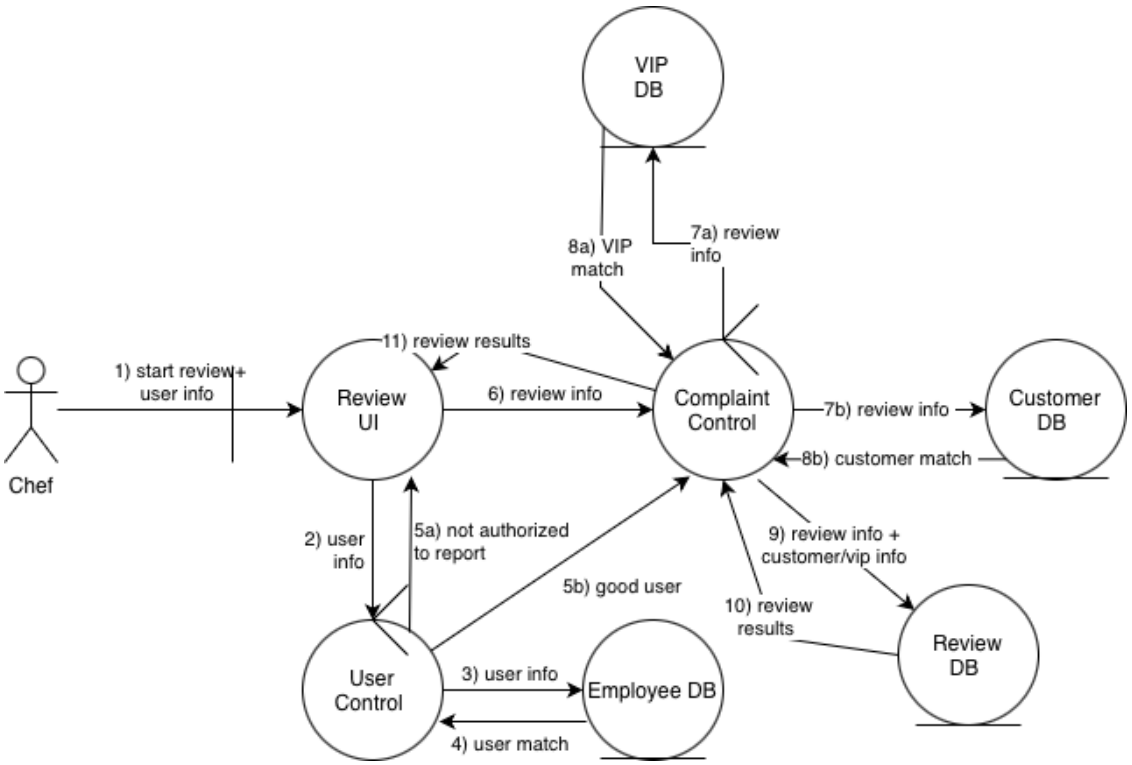
1. The chef searches for a customer.
2. The system fails to find a matching customer due to an invalid name, missing record, or the customer being recently deregistered.
3. The system displays an error message and prompts the chef to adjust the search or choose from recent customers.
4. The chef retries the search or cancels the complaint.

Exceptional Scenario B Incomplete Complaint Submission

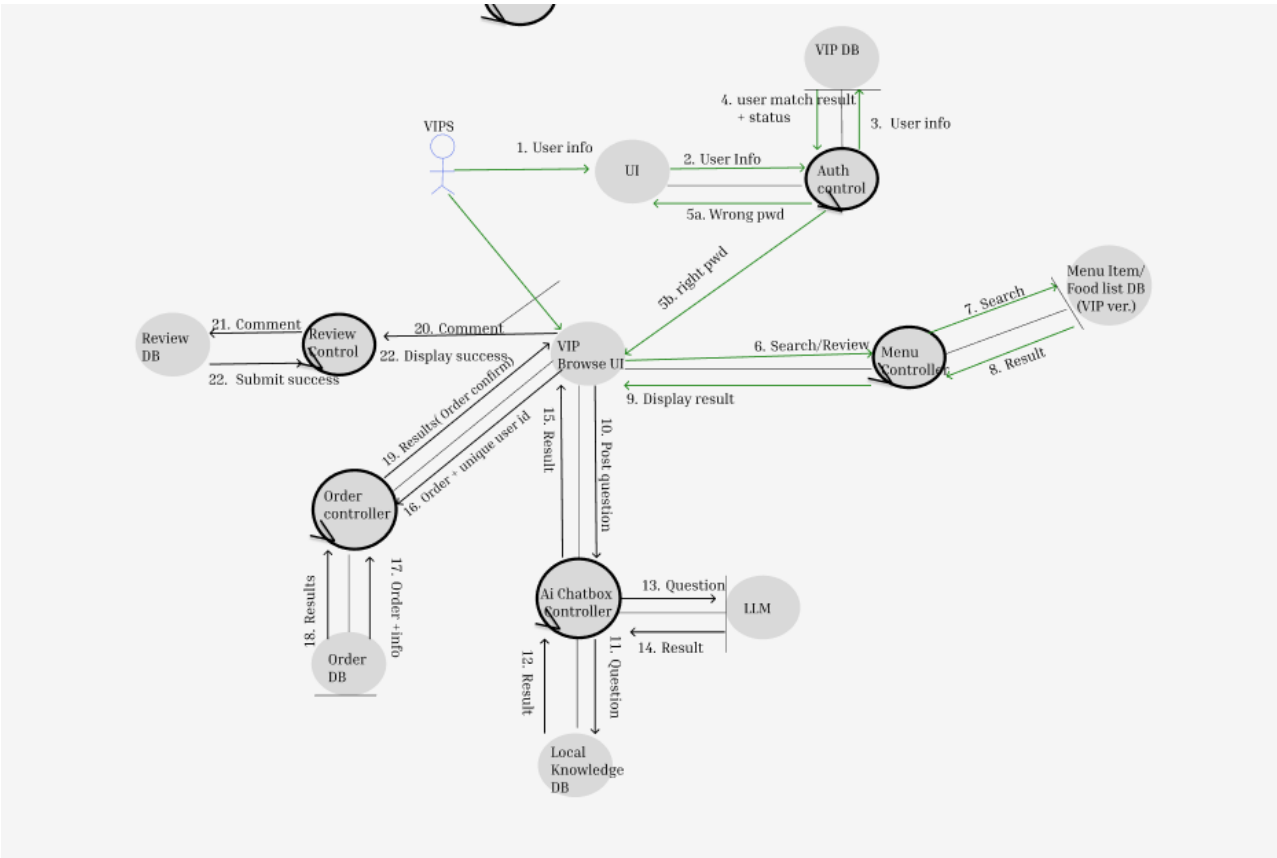
1. The chef fills out the complaint form but leaves required fields empty.
2. The system detects the missing information.
3. The system highlights the incomplete fields and prevents submission.
4. The chef completes the missing information and resubmits.

Exceptional Scenario C System Fails to Log Complaint

1. The chef submits the complaint.
2. A database or logging error occurs during submission.
3. The system rejects the submission and informs the chef of a temporary issue.
4. The chef may retry later, or the system may queue the complaint until the issue is resolved.


VIP
DB
8a) VIP
match
7a) review
info
11) review results
1) start review+
user info
Review
UI
6) review info
Complaint
Control
7b) review info
Customer
DB
8b) customer match
Chef
2) user
info
5a) not authorized
to report
9) review info +
customer/vip info
5b) good user
10) review
results
Review
DB
User
Control
3) user info
Employee DB
4) user match

VIP:



Normal Scenario

1. VIP Customer logs into the system.
2. The system updates the user interface tailored to VIPs for browsing.
3. If searching for a food item, the system checks the Menu item database tailored to VIPs, the system updates the UI with the displayed result(s).
4. If posting a question, the Ai chatbot refers to the local knowledge database first then returns its response to the user. If not found in local database, the Ai Chatbox refers to the LLM with the user's question as a prompt.
5. If the user posts a review/comment, the system adds it to the collection of reviews then returns a "success" output to the UI.
6. If the user places an order, the requested item along with the user id, gets sent to the system and updates it to the orders database. The system returns an "order confirmed" output to the screen.

Exceptional Scenario A — Failed Authorization

1. The user attempts to log in but enters the wrong user info or is no longer found in database due to being blacklisted by manager.

2. The system may prompt the user to reenter password before updating the interface or display blacklist status.

Exceptional Scenario B — Item not found

1. User attempts to search for an item that may be out of stock, not in database, or was never in database.
2. System displays “not found” or no items

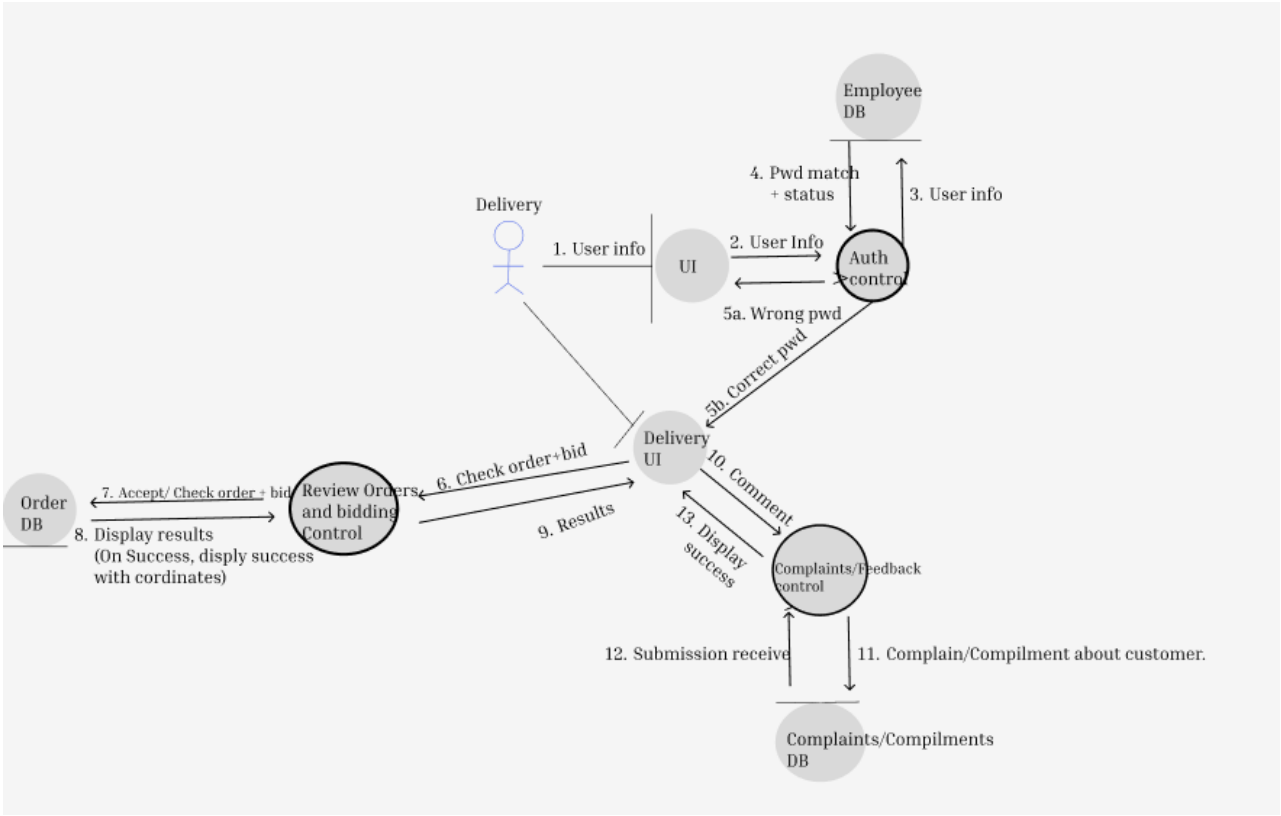
Exceptional Scenario C — Unsuccessful Transaction

1. Upon placing an order, system checks necessary requirements for the order to be placed (e.g. Sufficient or Insufficient funds)
2. System alerts the user order cannot be placed
3. User can re-enter required info for checkout UI

Exceptional Scenario D — Rating/Commenting without history of purchase

1. User attempts to rate/comment on a order.
2. The system checks if this order id exists.
3. The system rejects the submission on failed attempt to find order id.

Delivery person:



Normal Scenario

1. The delivery person logs into the system with correct user info that matches the employee database.
2. The system returns UI tailored for delivery people.
3. The delivery person views the current orders.
4. The delivery person is able to place a bid on an order to be reviewed by manager. On success, the system displays confirmation along with the coordinates to the order.
5. The user can send complaints/compliments/feedback about a customer along with the customer's id to the complaint/compliment DB.
6. The system returns a "successful submission" output to the screen.

Exceptional Scenario A — Failed Authorization

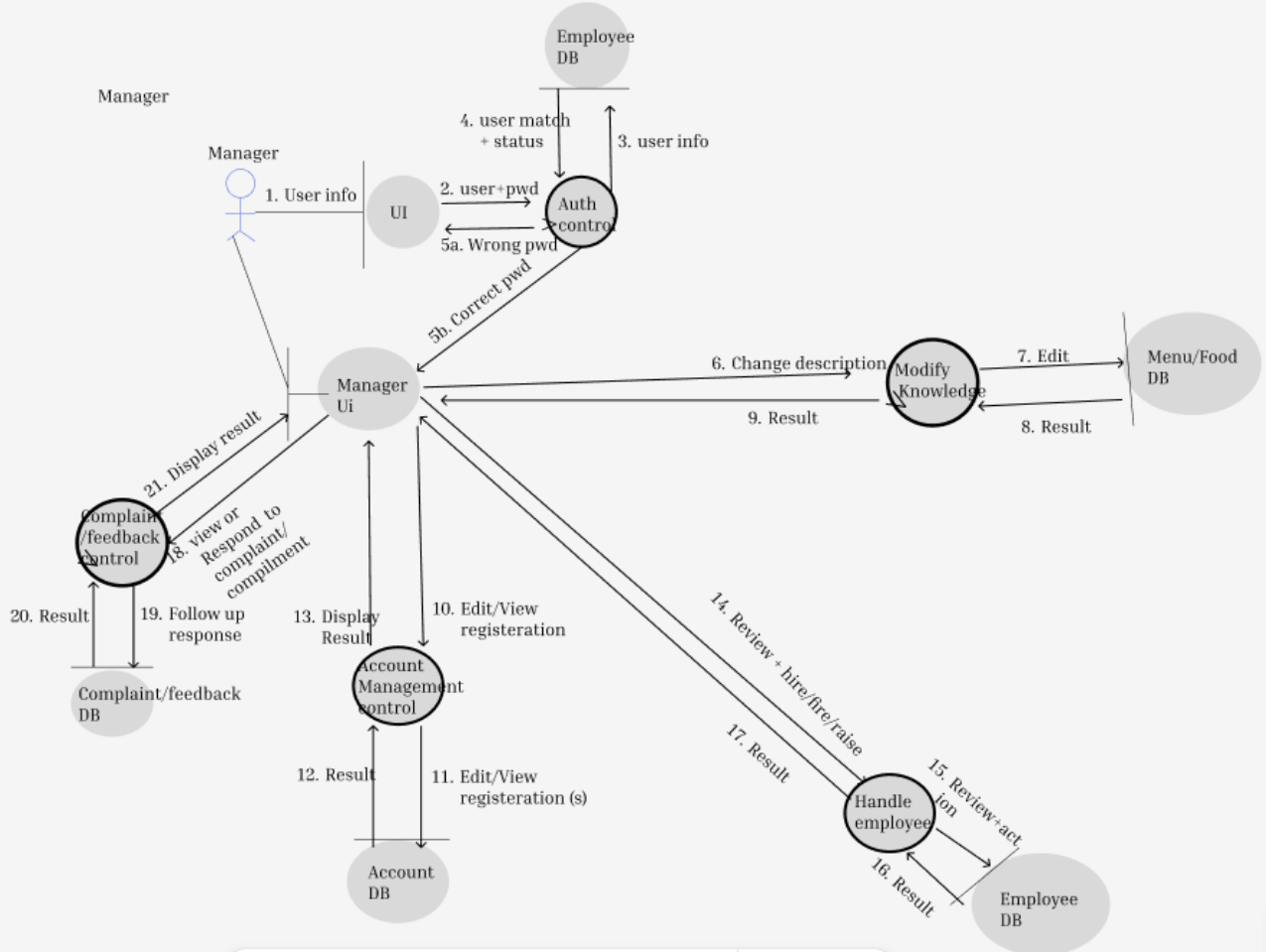
1. The user attempts to log in.
2. The user inputs wrong info or received too many complaints prior which is logged in the employee database.
3. The system either re-prompts the user with unsuccessful log in alert. or show current employee status

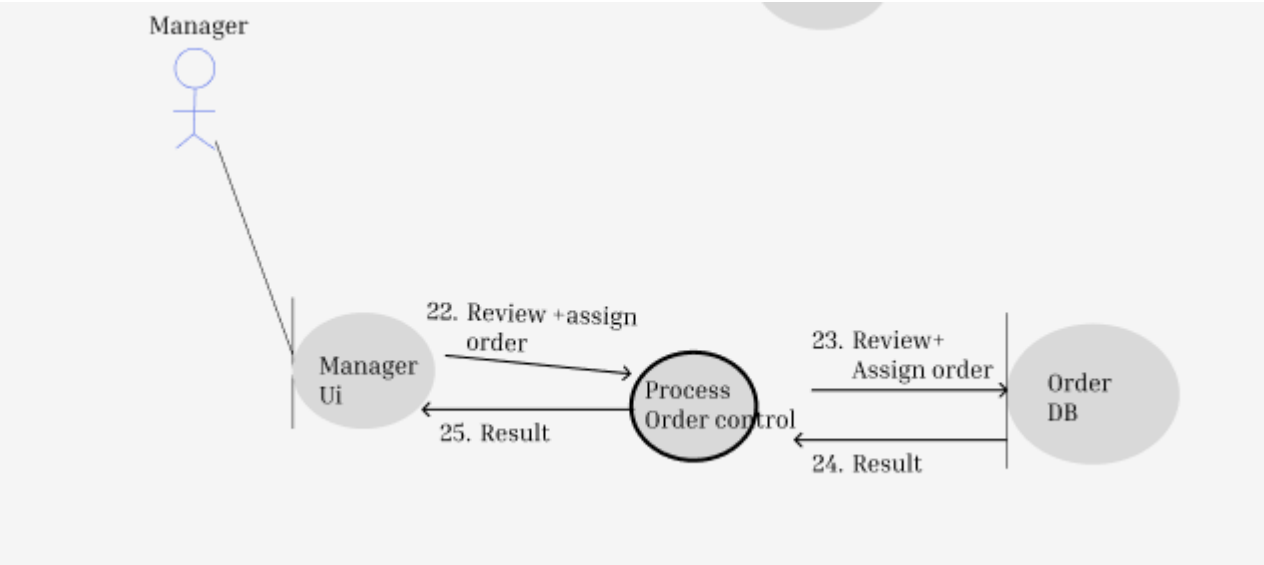
Exceptional Scenario B — Denied bid

1. The user requests to take on an order with allocated bid amount.

2. The manager rejects bid offer and allocates the order to a different bidder.
3. The system displays the results. (e.g. “order no longer available”)

Manager:





Normal Scenario:

1. The user starts by logging into the system which in turn displays the manager interface.
2. The user has the option to update knowledge regarding the restaurant menu, including prices, descriptions, images etc. (Similar to the chef I'm assuming since they are the manager). Once the manager inputs the change into the database, the change is immediately reflected on the screen.
3. If the user wishes to handle customer registrations, the system will retrieve a list of customers to view with options to help handle the registration/deregistration.
4. In addition, the system can access and retrieve a set of employee accounts, and the user has the option to provide a raise/hire/fire them. (Assuming the user base their descionns on the amount of complaints/compliments)
5. Manager Ui allows the user to view complaints/compliments. The system allows the user to view a list of complaints/compliments from the database. The user can choose to respond to these as well.
6. The user can review all pending orders and tailored bids for these orders and approve them, which the system then notifies the delivery people of this.

Exceptional Scenario A — Failed Authorization

1. The user attempts to log in with the wrong info.
2. The user gets prompted to re-enter info.

Exceptional Scenario B— Modifying Menu

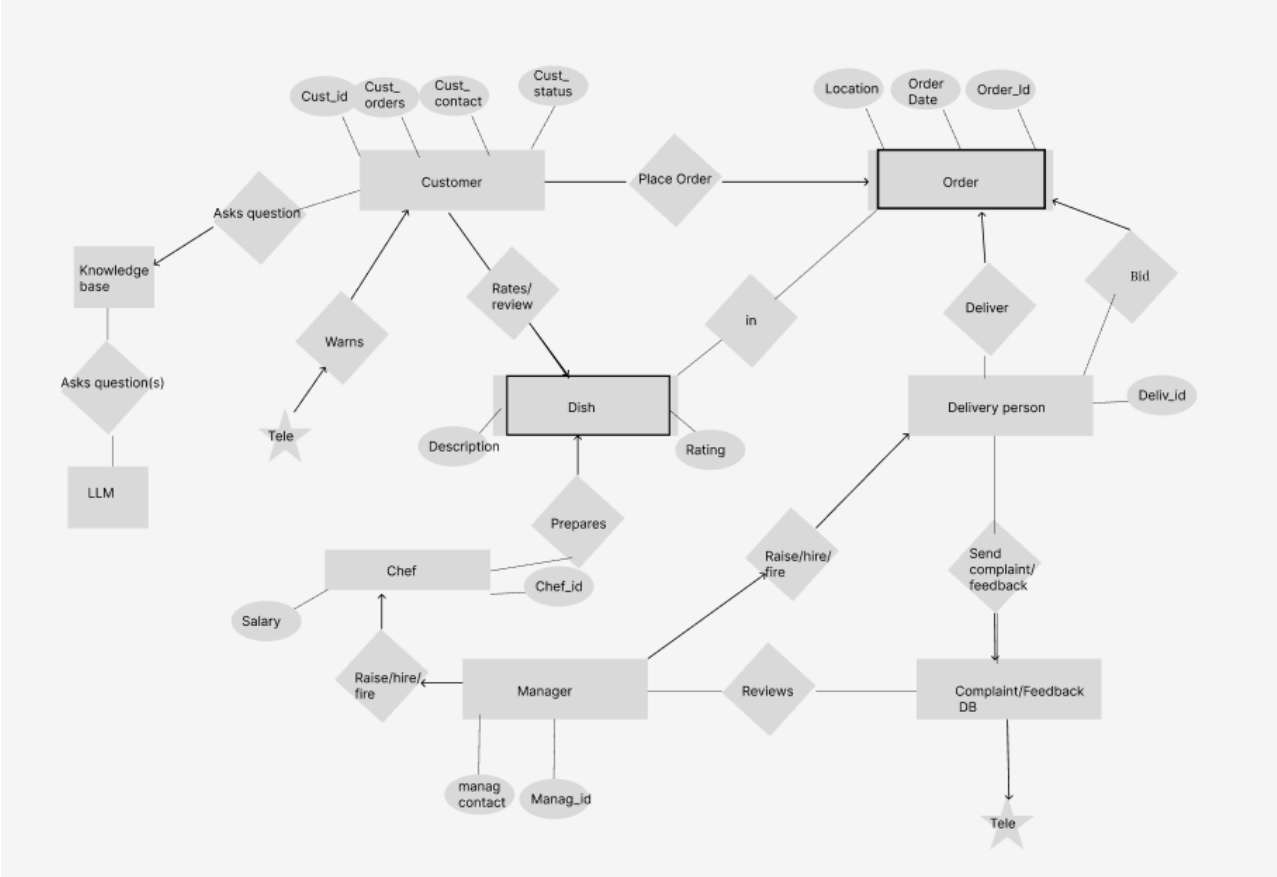
1. The user attempts to change the details of the menu.

2. The menu choice is no longer available. (e.g. Chef deletion)
3. The system alerts the user that the item is no longer available.

Exceptional Scenario C — No current bids placed

1. The user reviews current orders.
2. If no bids placed at all, the user can’t perform assignments until a bid has been placed.

Entity Relationship Diagrams



Detailed Design

Order Food - Methods

Component: UI => Method: receiveLoginCredentials(userInfo, pwd)
`INPUT: userInfo, pwd`
`OUTPUT: forward to Auth or error message`
`BEGIN`
`  SEND pwd TO Auth.authenticate(pwd, userInfo)`
`END`
	Component: UI => Method: displayWrongPassword()
	`INPUT: none`
	`OUTPUT: UI message`
	`BEGIN`
	`  SHOW "Incorrect password"`
	`END`

Component: Auth => Method: authenticate(pwd, userInfo)	Component: PWD DB => Method: fetchHashedPassword(userInfo)

```
INPUT: pwd, userInfo
OUTPUT: authentication result (true/false)

BEGIN
    SEND pwd TO PWD_DB.fetchHashedPassword(userInfo)

    receivedHash ← WAIT_FOR PWD_DB response

    IF match(pwd, receivedHash) THEN
        RETURN true
    ELSE
        RETURN false
    ENDIF
END
```

```
INPUT: userInfo
OUTPUT: hashedPassword

BEGIN
    hashedPassword ← QUERY database WHERE user = userInfo
    RETURN hashedPassword
END
```

Component: Checkout UI => Method: receiveFoodSelection(foodInfo, deliveryInfo)

```
INPUT: foodInfo, deliveryInfo
OUTPUT: forward to OrderFood

BEGIN
    SEND (foodInfo, deliveryInfo) TO OrderFood.placeOrder()
END
```

Component: Order Food=> Method: placeOrder(foodInfo, deliveryInfo)

```
INPUT: foodInfo, deliveryInfo
OUTPUT: orderConfirmation

BEGIN
    SEND foodInfo TO OrdersDB.storeOrder(foodInfo, deliveryInfo)

    result ← WAIT_FOR OrdersDB response

    RETURN result
END
```

Component: Orders DB => Method: storeOrder(foodInfo, deliveryInfo)

```
INPUT: foodInfo, deliveryInfo
OUTPUT: confirmation / orderID

BEGIN
    orderID ← INSERT INTO Orders(foodInfo, deliveryInfo)

    RETURN "Order Confirmed: " + orderID
END
```

Ask Questions (AI Query System)

Component: AI UI	Component: AI UI

=> Method: submitPrompt(prompt)

```
INPUT: prompt
OUTPUT: AI response

BEGIN
    SEND prompt TO AI_Control.processPrompt(prompt)
END
```

=> Method: displayResponse(response)

```
INPUT: response
OUTPUT: UI display

BEGIN
    SHOW response ON screen
END
```

Component: AI Control

=> Method: processPrompt(prompt)

```
INPUT: prompt
OUTPUT: finalResponse

BEGIN
    SEND prompt TO LocalKnowledgeDB.lookup(prompt)

    localInfo ← WAIT_FOR LocalKnowledgeDB response

    IF localInfo contains high-confidence answer THEN
        SEND (prompt, localInfo) TO LocalAI.generateResponse()
        response ← WAIT_FOR LocalAI
    ELSE
        SEND (prompt, localInfo) TO ExternalAI.query()
        response ← WAIT_FOR ExternalAI
    ENDIF

    RETURN response
END
```

Component: Local Knowledge DB

=> Method: lookup(prompt)

```
INPUT: prompt
OUTPUT: infoAboutPrompt

BEGIN
    info ← QUERY LocalKnowledge WHERE keywords match prompt
    RETURN info
END
```

Component: Local AI

=> Method: generateResponse(prompt, info)

```
INPUT: prompt, info
OUTPUT: AI-generated response

BEGIN
    response ← PROCESS prompt USING info
    RETURN response
END
```

Component: External AI

=> Method: query(prompt, info)

```
INPUT: prompt, info
OUTPUT: response

BEGIN
    response ← CALL external AI API(prompt, info)
    RETURN response
END
```

Review Order— Methods

Component: Review UI

=> Method: startReview(userInfo)

```
INPUT: userInfo
OUTPUT: forward to ReviewControl

BEGIN
    SEND userInfo TO ReviewControl.verifyUser(userInfo)
END
```

Component: Review UI

=> Method: displayNotAllowed()

```
INPUT: none
OUTPUT: UI message

BEGIN
    SHOW "You are not allowed to review this order."
END
```

Component: Review UI

=> Method: verifyUser(userInfo)

```
INPUT: userInfo
OUTPUT: allow / deny

BEGIN
    SEND userInfo TO CustomerDB.lookupUser()
    resultCustomer ← WAIT_FOR response

    SEND userInfo TO VIP_DB.lookupVIP()
    resultVIP ← WAIT_FOR response

    IF resultCustomer = false AND resultVIP = false THEN
         RETURN "not allowed"
    ENDIF

    RETURN "allowed"
END
```

Component: Review UI

=> Method: fetchOrderInfo(orderID, userInfo)

```
INPUT: orderID, userInfo
OUTPUT: order details

BEGIN
    SEND (orderID, userInfo) TO OrderDB.getOrder()
    orderInfo ← WAIT_FOR response

    RETURN orderInfo
END
```

Component: Review UI

=> Method: submitReview(reviewInfo, orderInfo)

```
INPUT: reviewInfo, orderInfo
OUTPUT: confirmation

BEGIN
    SEND (reviewInfo, orderInfo) TO ReviewDB.storeReview()

    result ← WAIT_FOR response

    RETURN result
END
```

Component: Review DB

=> Method: storeReview(reviewInfo, orderInfo)

```
INPUT: reviewInfo, orderInfo
OUTPUT: success message

BEGIN
    INSERT review INTO ReviewTable

    RETURN "Review Submitted Successfully"
END
```

Remove / Add Dishes (Chef)

Component: UI (Chef Menu Editor)

=> Method: submitMenuModification(modification, userInfo)

```
INPUT: modification, userInfo
OUTPUT: success / denied

BEGIN
    SEND (modification, userInfo) TO BrowseControl.processModification()
END
```

Component: Browse Control

=> Method: processModification(modification, userInfo)

```
INPUT: modification, userInfo
OUTPUT: result

BEGIN
    SEND userInfo TO EmployeeDB.verifyUser()
    valid ← WAIT_FOR response

    IF valid = false THEN
        RETURN "Not authorized"
    ENDIF

    SEND modification TO FoodDB.applyModification()
    result ← WAIT_FOR FoodDB response

    RETURN result
END
```

Component: Employee DB

=> Method: verifyUser(userInfo)

```
INPUT: userInfo
OUTPUT: true/false

BEGIN
    match ← QUERY employees WHERE user = userInfo
    RETURN match != NULL
END
```

Component: Food DB

=> Method: applyModification(modification)

```
INPUT: modification (add/remove/update)
OUTPUT: operation result

BEGIN
    IF modification.type = "add" THEN
        INSERT new dish
    ELSE IF modification.type = "remove" THEN
        DELETE dish
    ELSE IF modification.type = "update" THEN
        UPDATE dish fields
    ENDIF

    RETURN "Modification Applied"
END
```

Complain About Customers

Component: Review UI (Chef)

=> Method: startComplaint(userInfo)

Component: User Control

=> Method: verifyEmployee(userInfo)

```
INPUT: userInfo
OUTPUT: forward to UserControl

BEGIN
    SEND userInfo TO UserControl.verifyEmployee()
END
```

```
INPUT: userInfo
OUTPUT: allow / deny

BEGIN
    SEND userInfo TO EmployeeDB.verifyUser()
    result ← WAIT_FOR response

    IF result = false THEN
        RETURN "not authorized"
    ENDIF

    RETURN "good user"
END
```

Component: Complaint Control

=> Method: processComplaint(reviewInfo)

```
INPUT: reviewInfo
OUTPUT: confirmation

BEGIN
    SEND reviewInfo TO CustomerDB.lookupCustomer()
    customer ← WAIT_FOR response

    SEND reviewInfo TO VIP_DB.lookupVIP()
    vip ← WAIT_FOR response

    combinedInfo ← MERGE(customer, vip, reviewInfo)

    SEND combinedInfo TO ReviewDB.storeComplaint()

    result ← WAIT_FOR ReviewDB response

    RETURN result
END
```

Component: Review DB

=> Method: storeComplaint(info)

```
INPUT: info
OUTPUT: confirmation

BEGIN
    INSERT complaint INTO ComplaintTable

    RETURN "Complaint Stored"
END
```

Detailed Design

System Screens

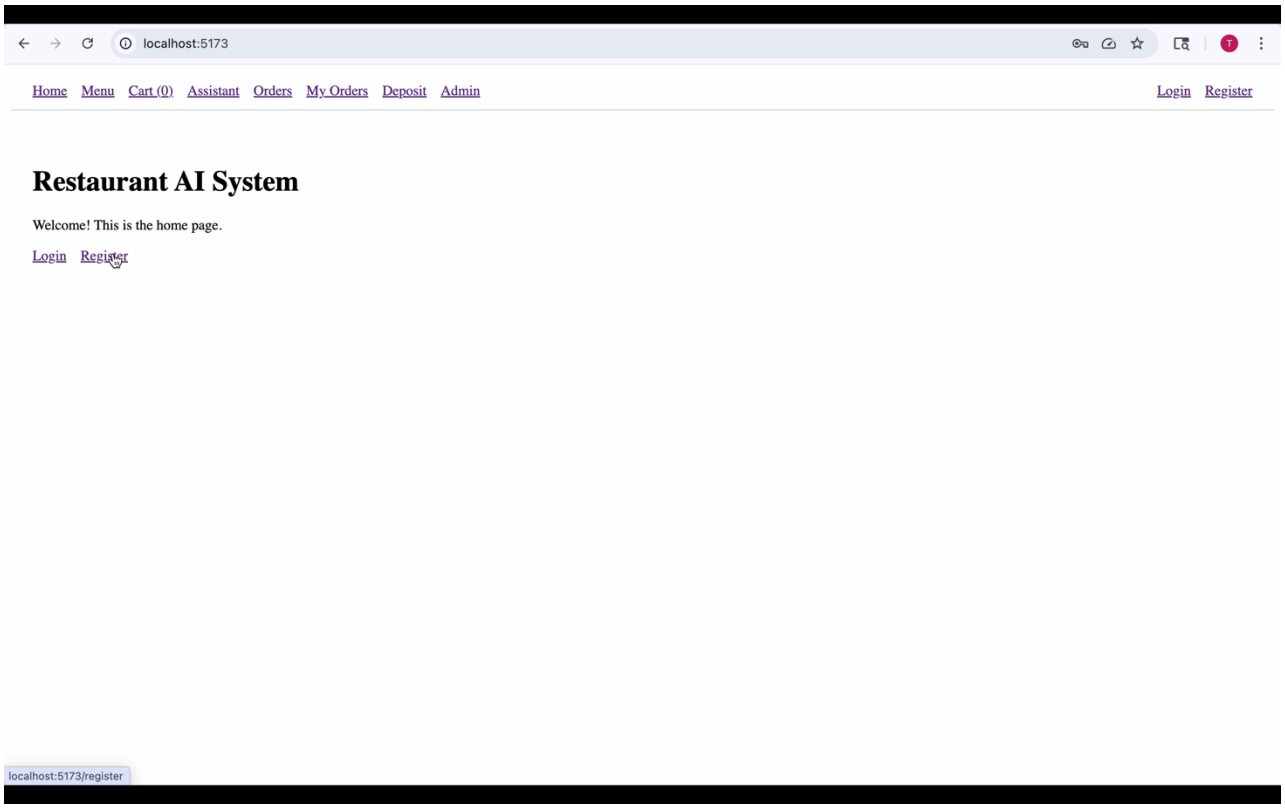
1. Landing Page

Purpose:

The Landing Page is the first page users see when they open the application. It introduces the system and provides quick access to essential navigation links

Main Features:

- Top navigation bar with links to **Home**, **Menu**, **Cart**, **Orders**, **My Orders**, **Deposit**, **Assistant**, and **Admin** (if authorized).
- Display of the system title: *Restaurant AI System*.
- Simple welcome message for first-time visitors.
- Quick-access links to **Login** and **Register**.
- If a user is logged in, navigation options dynamically update to show user-specific operations.



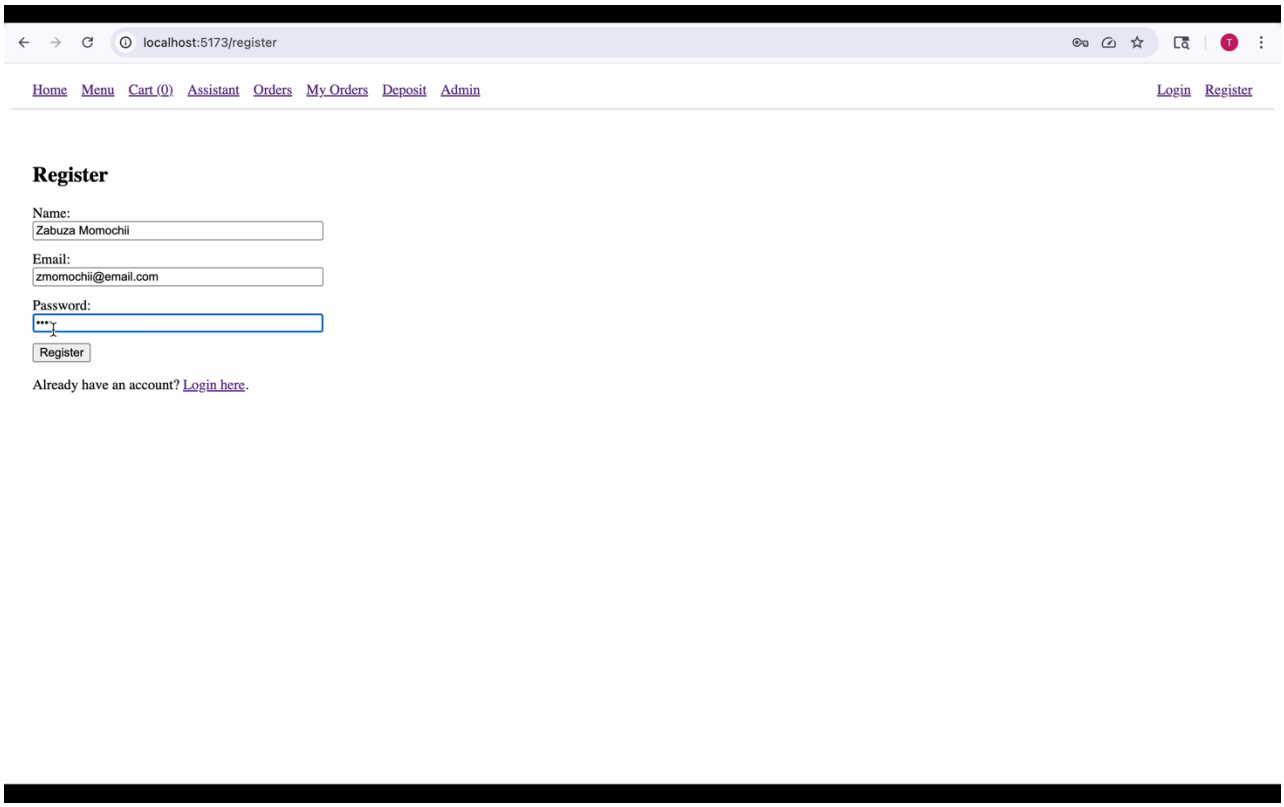
2. Registration Page

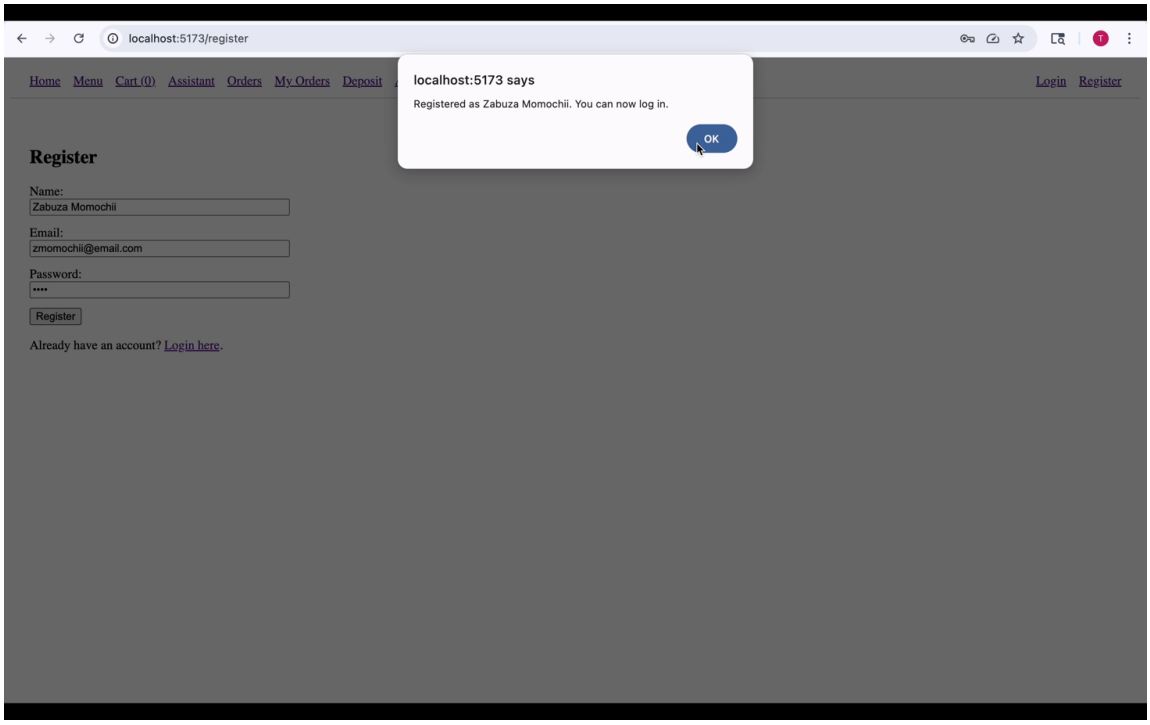
Purpose:

Allows new users to create an account in order to access the system's features like placing orders and managing their cart.

Main Features:

- Input fields: **Name**, **Email**, **Password**, and any additional required fields.
- Form validation to ensure correct input formats.
- Button: **Register** to create the account.
- Link to **Login** for users who already have an account.
- Error messages for duplicate email or invalid inputs.





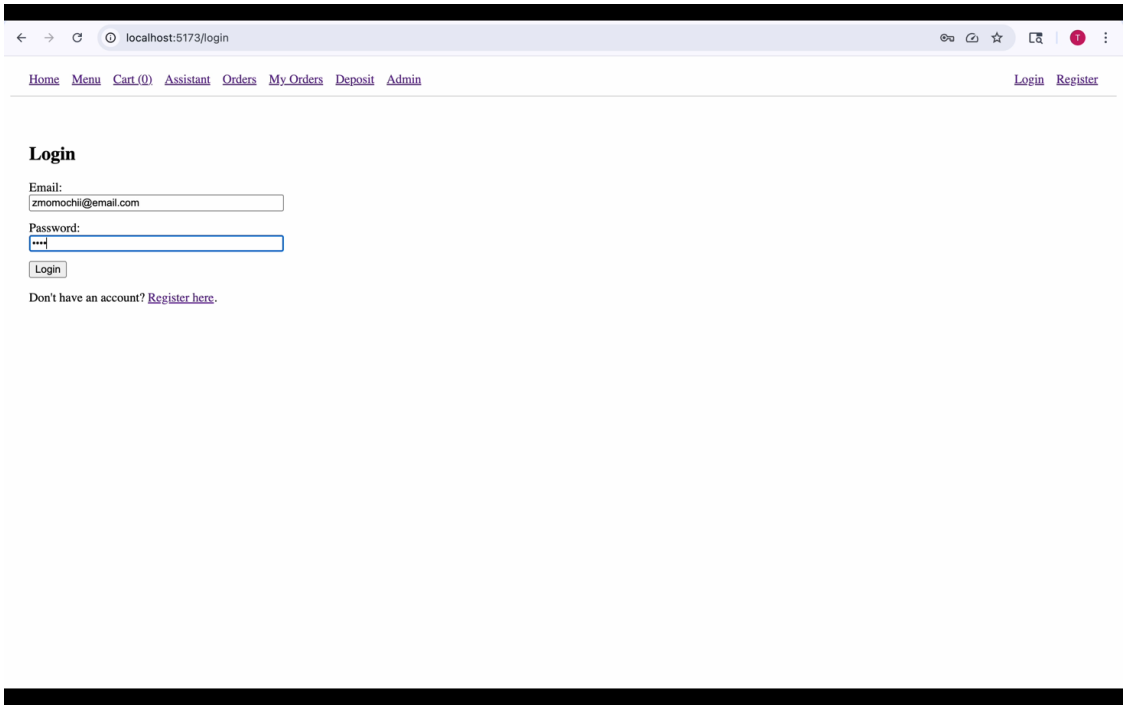
3. Login Page

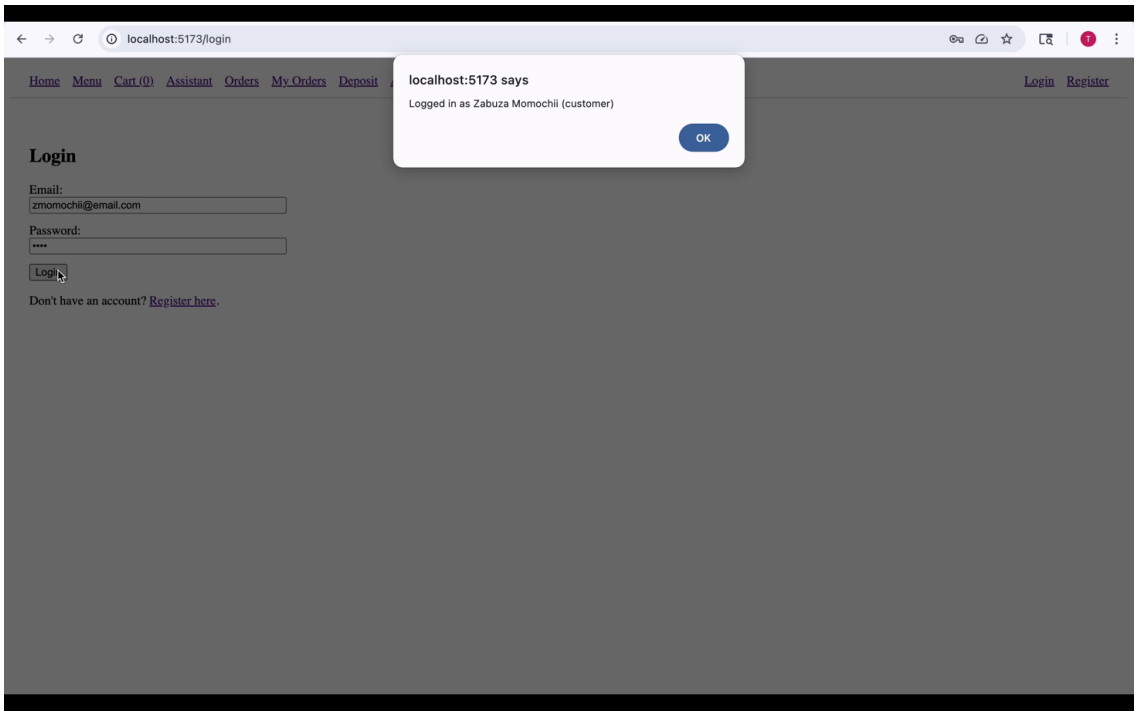
Purpose:

Allows existing users to authenticate and access their personalized dashboard, cart, and order history.

Main Features:

- Input fields: **Email**, **Password**.
- Button: **Login**.
- Link to **Register** if the user doesn't have an account.
- Error messages for incorrect credentials.





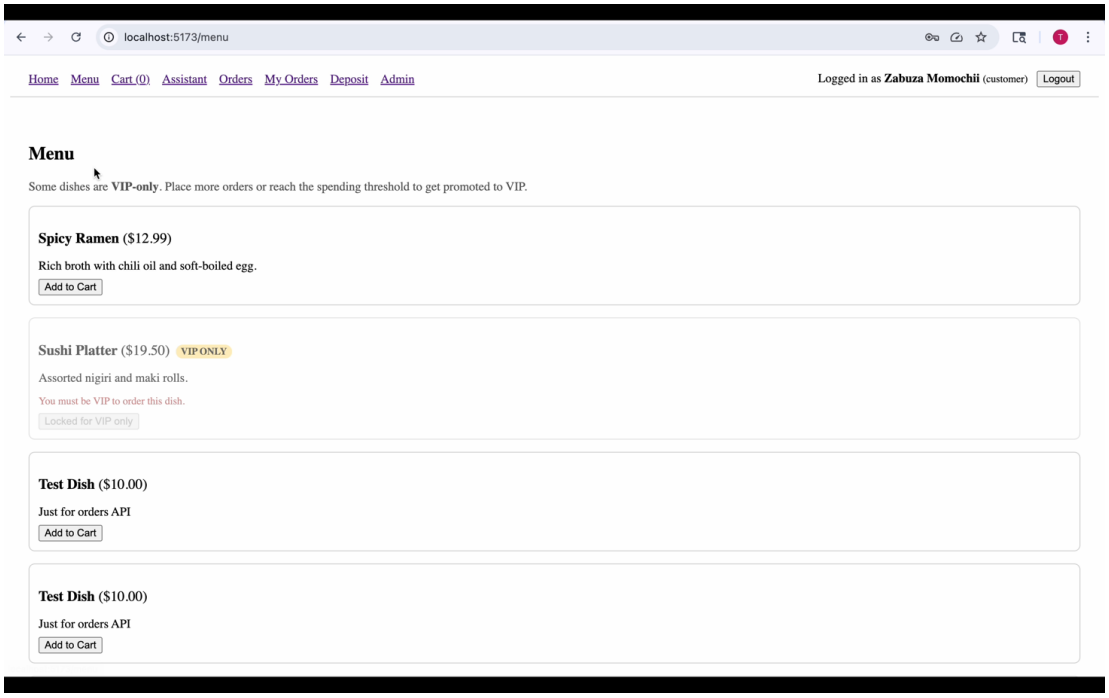
4. Menu Page

Purpose:

Displays all available restaurant items for browsing and ordering.

Main Features:

- List/grid of menu items with **name**, **description**, **price**, and **Add to Cart** button.
- Filters or categories (if implemented).
- Clicking an item may show item details.
- Visible cart count updates dynamically.



5. Cart Page

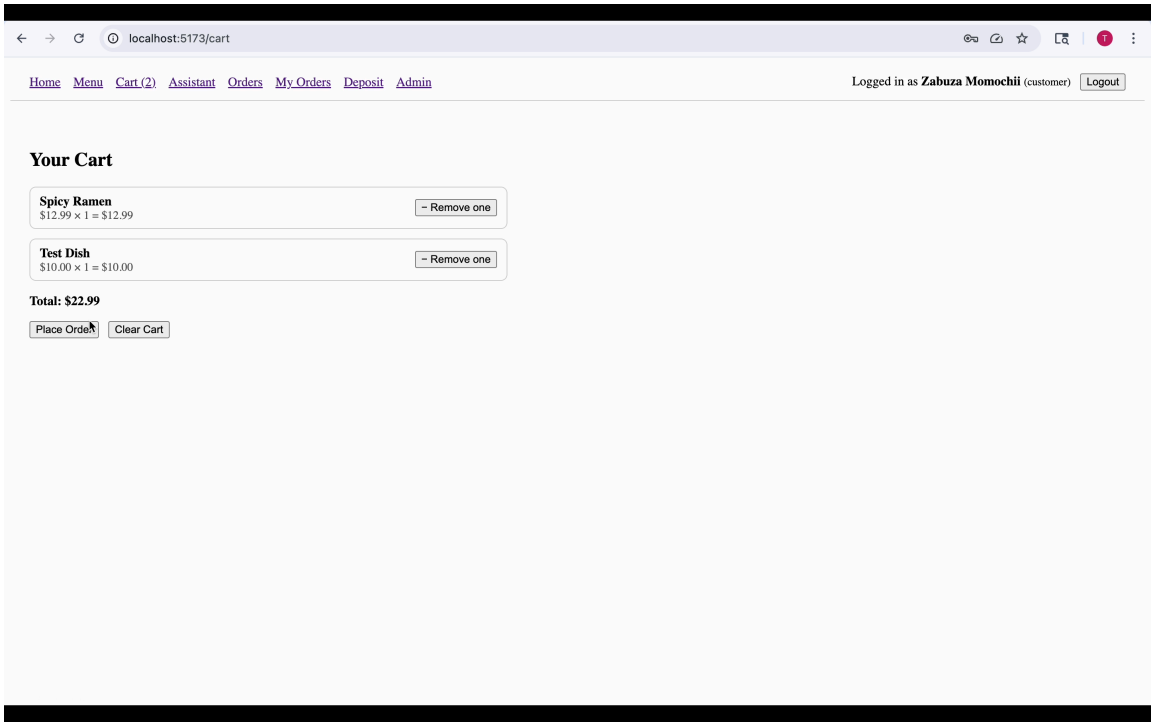
Purpose:

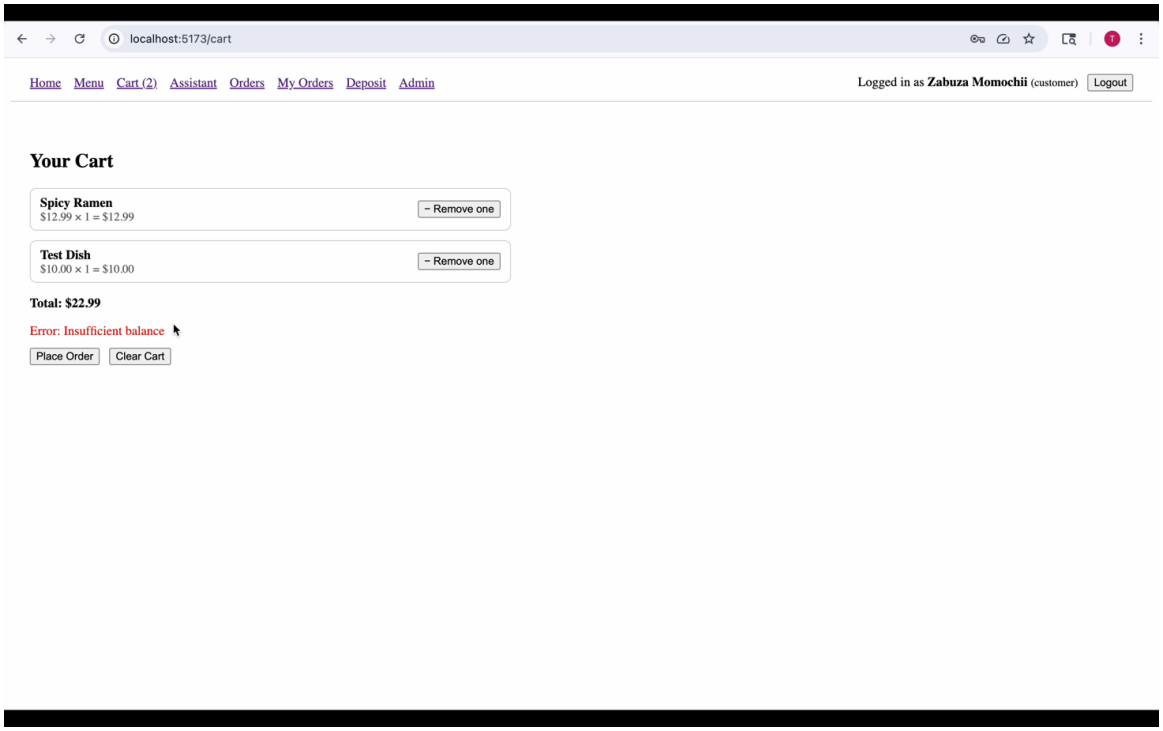
Allows users to review, update, or remove items from their shopping cart before placing an order.

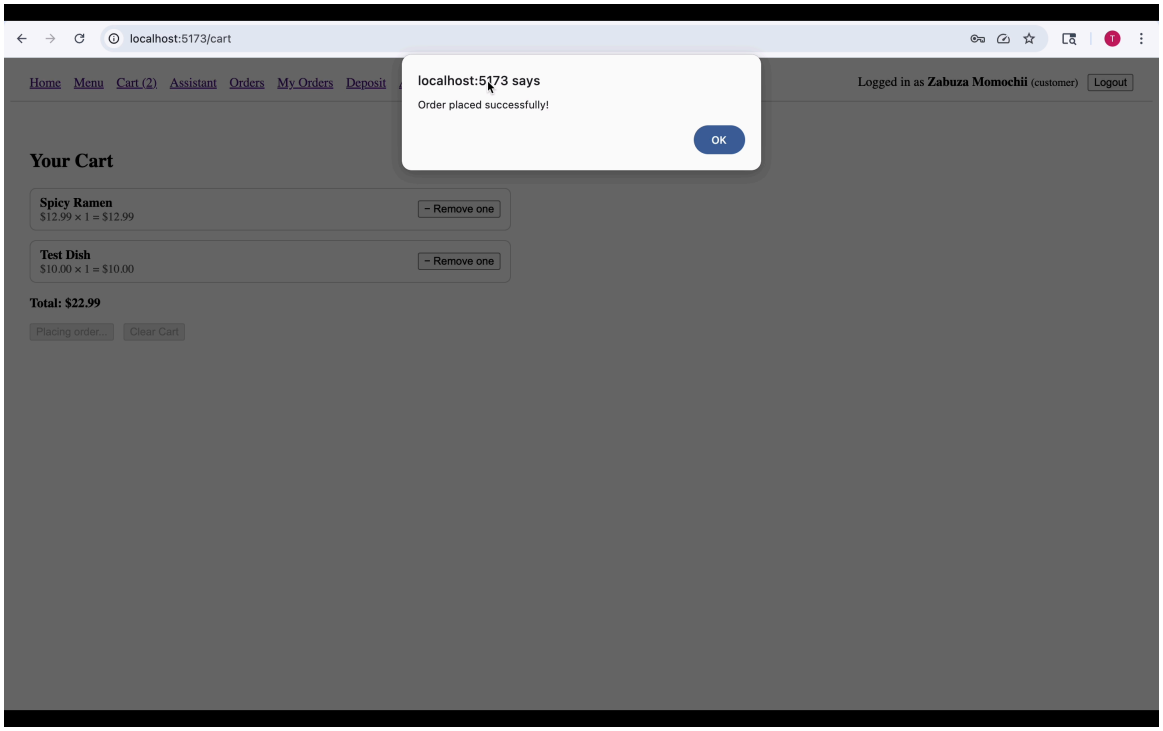
Main Features:

- List of items added to cart with **quantity** and **price**.

- Buttons for **Update Quantity**, **Remove**, or **Clear Cart**.
- Total price calculation.
- Button to proceed to **Checkout / Place Order**.







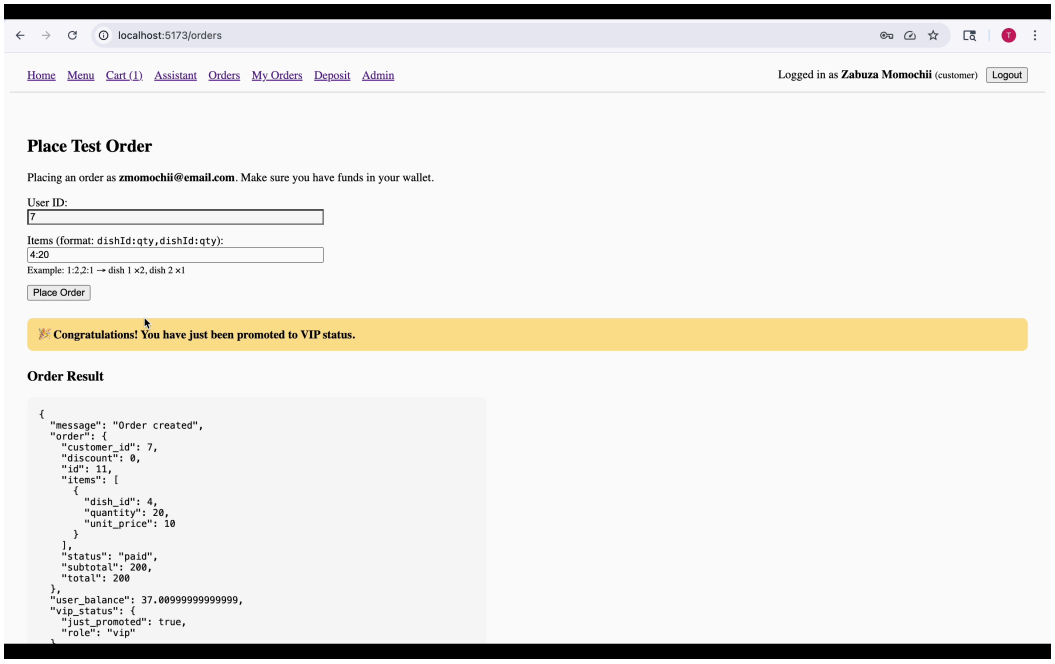
6. Checkout / Place Order Page

Purpose:

Collects final information needed to confirm the order.

Main Features:

- Displays order summary: items, quantities, total cost.
- Delivery details or payment selection (if implemented).
- Button: **Confirm Order**.
- Validations to ensure cart is not empty and items are still available.



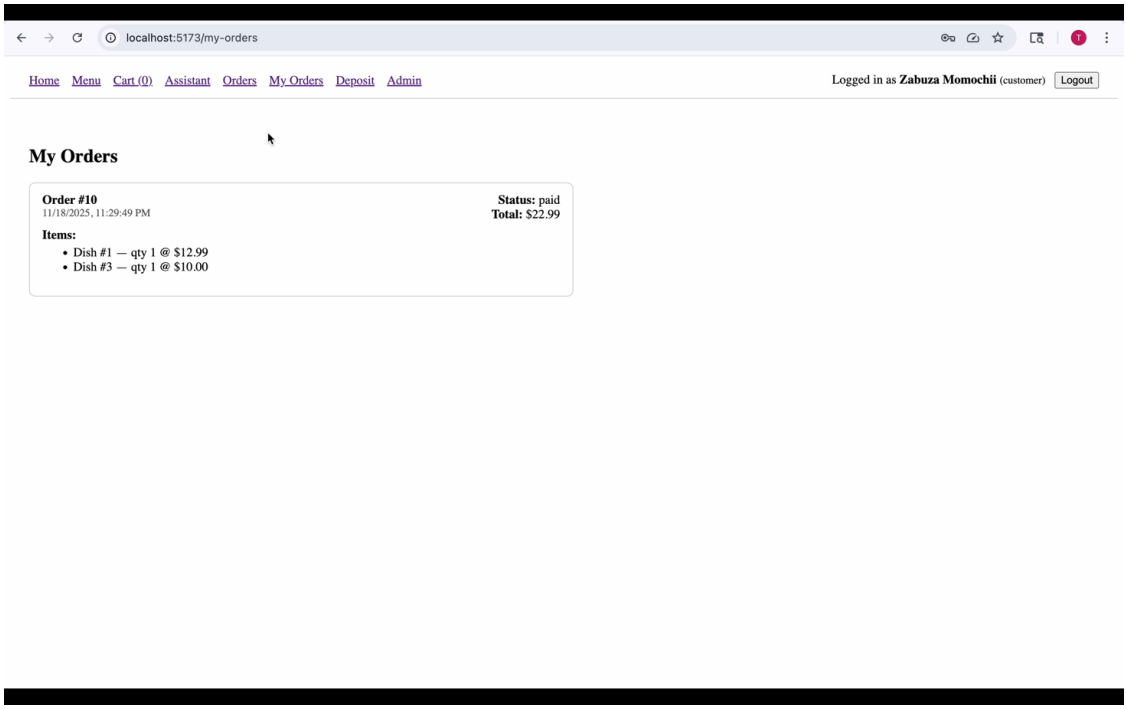
7. My Orders (Order History) Page

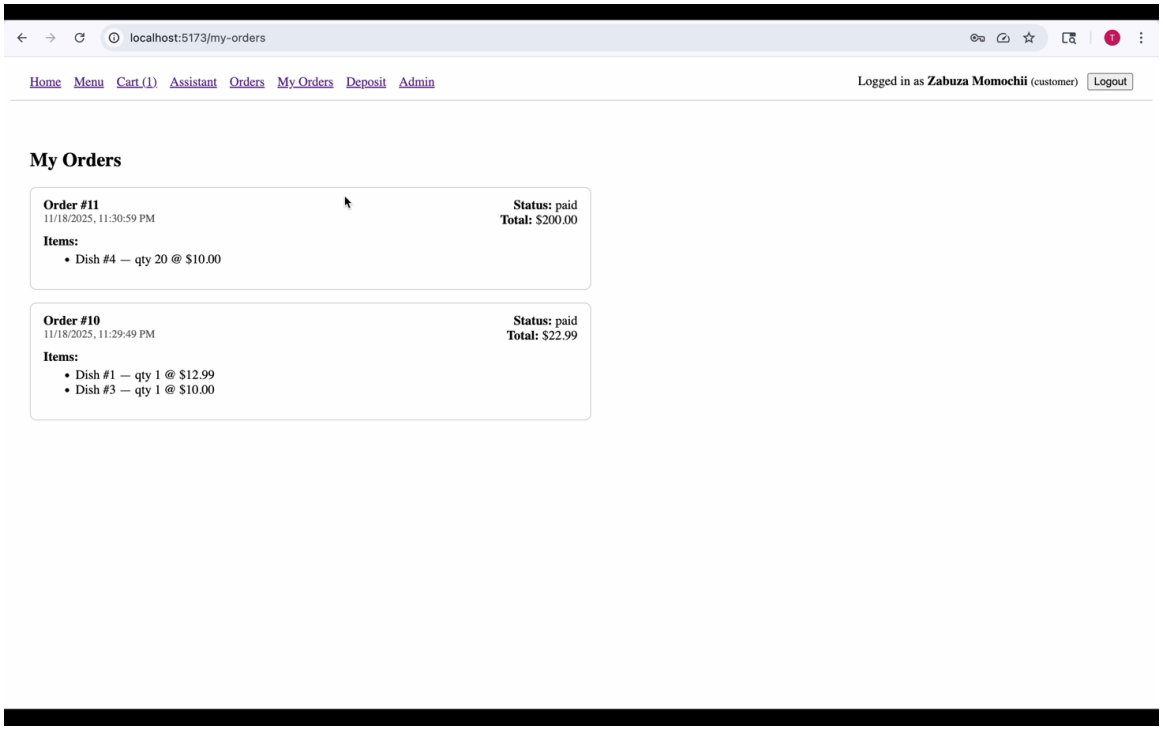
Purpose:

Shows all past and current orders made by the user.

Main Features:

- List of orders with **Order ID**, **Date**, **Status**, **Total Amount**.
- Clickable links to open detailed order view.





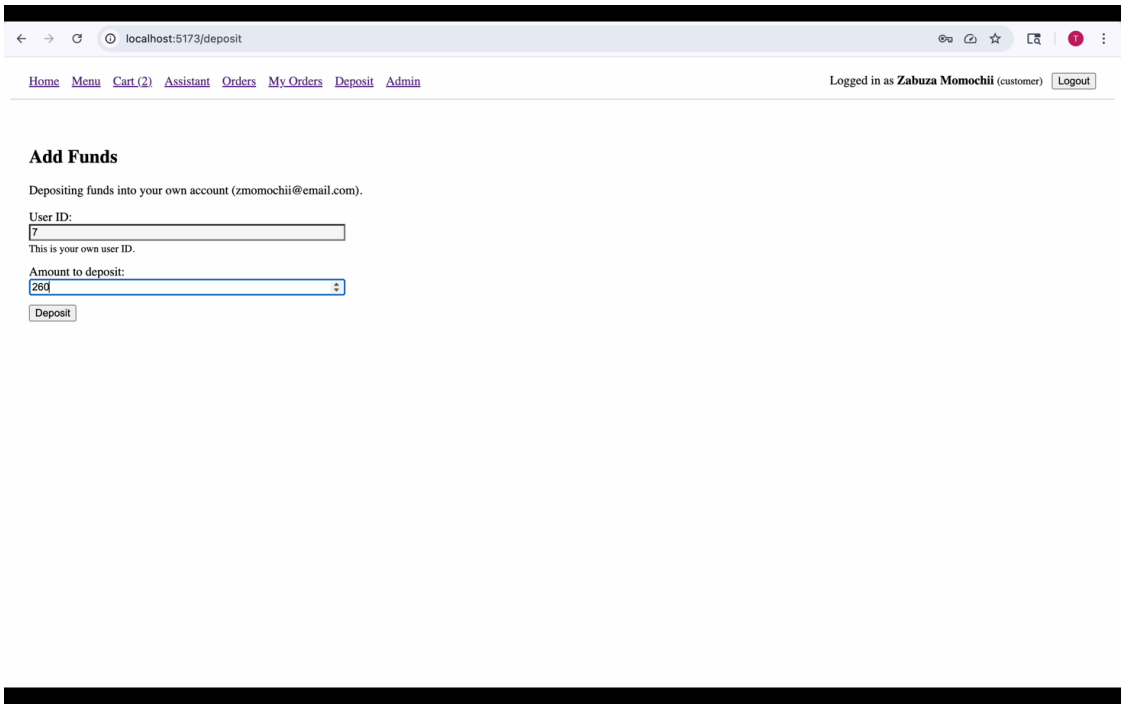
8. Deposit Page

Purpose:

Allows users to add funds or credits to their account balance (if your team implemented it).

Main Features:

- Input for deposit amount.
- Button to **Submit Payment** or Add Credit.
- Confirmation message.



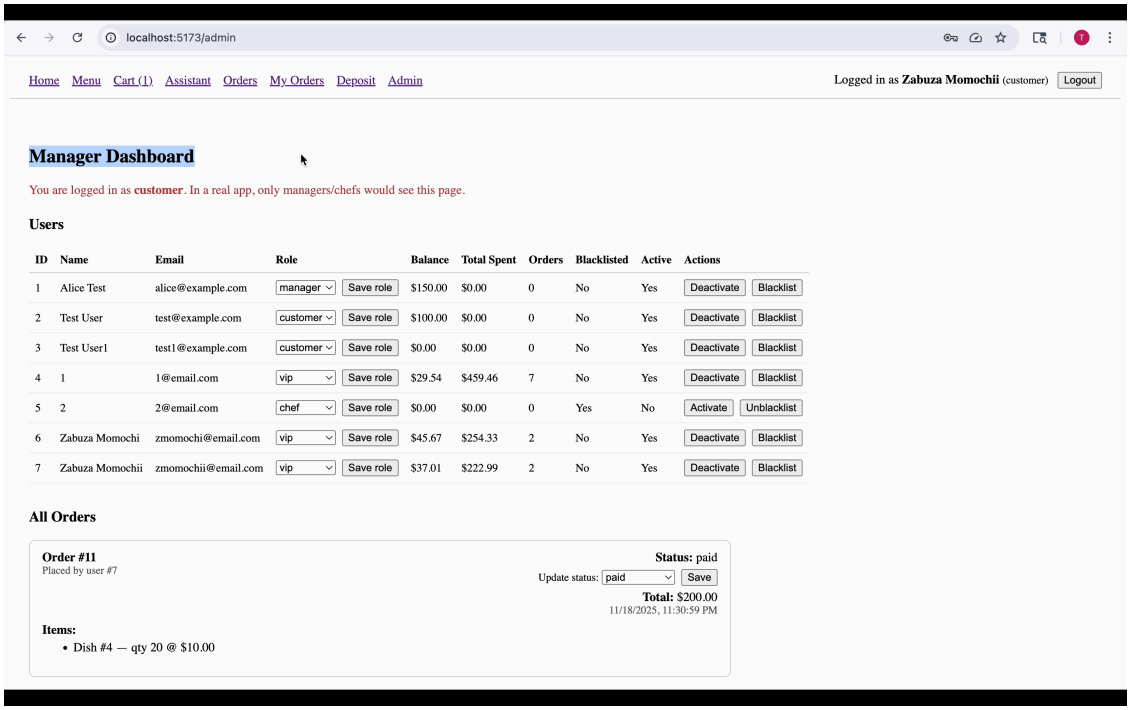
9. Admin Dashboard Page

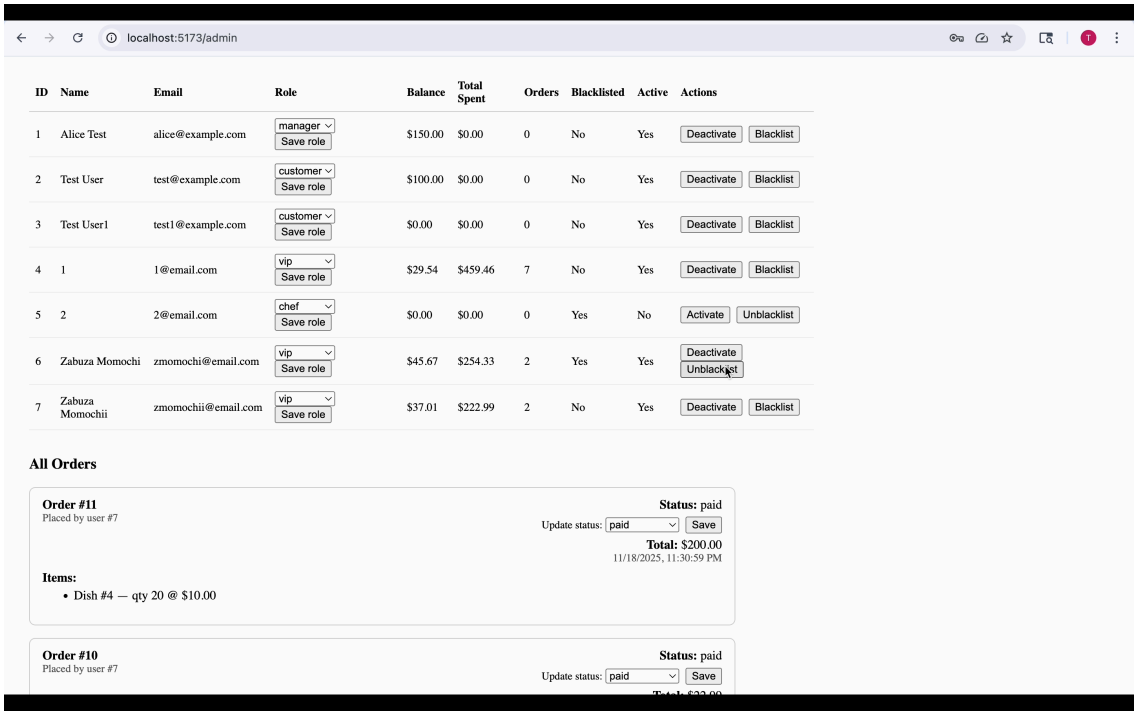
Purpose:

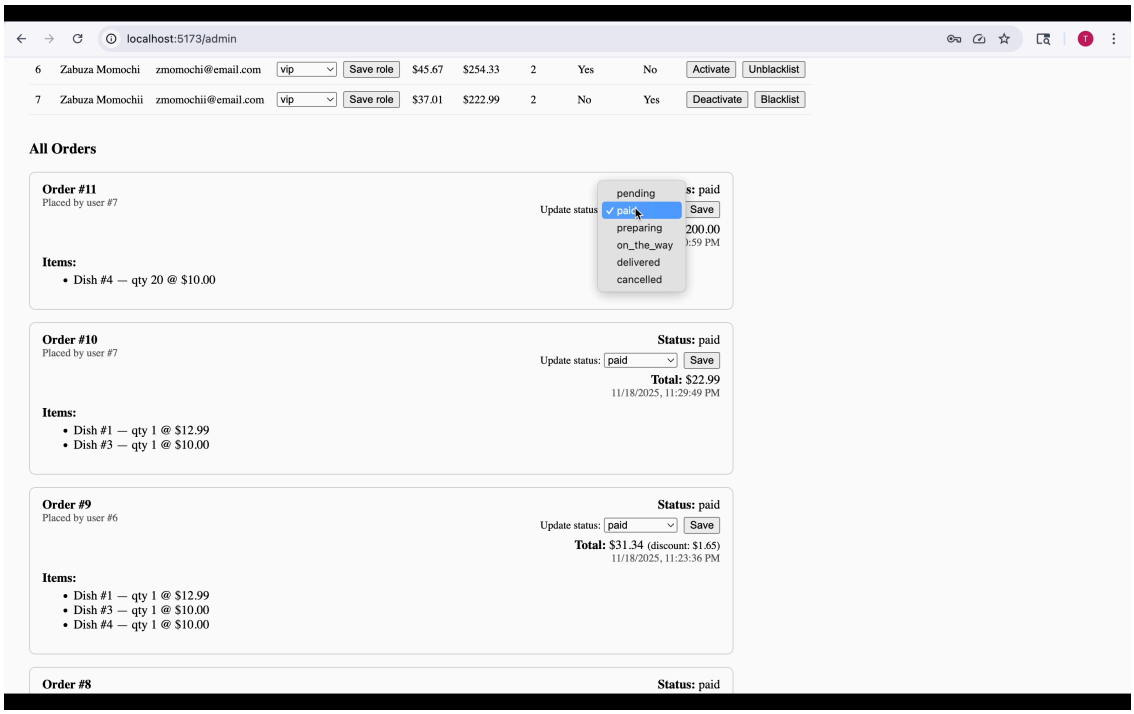
Only accessible to admins. Allows management of menu items, orders, and system data.

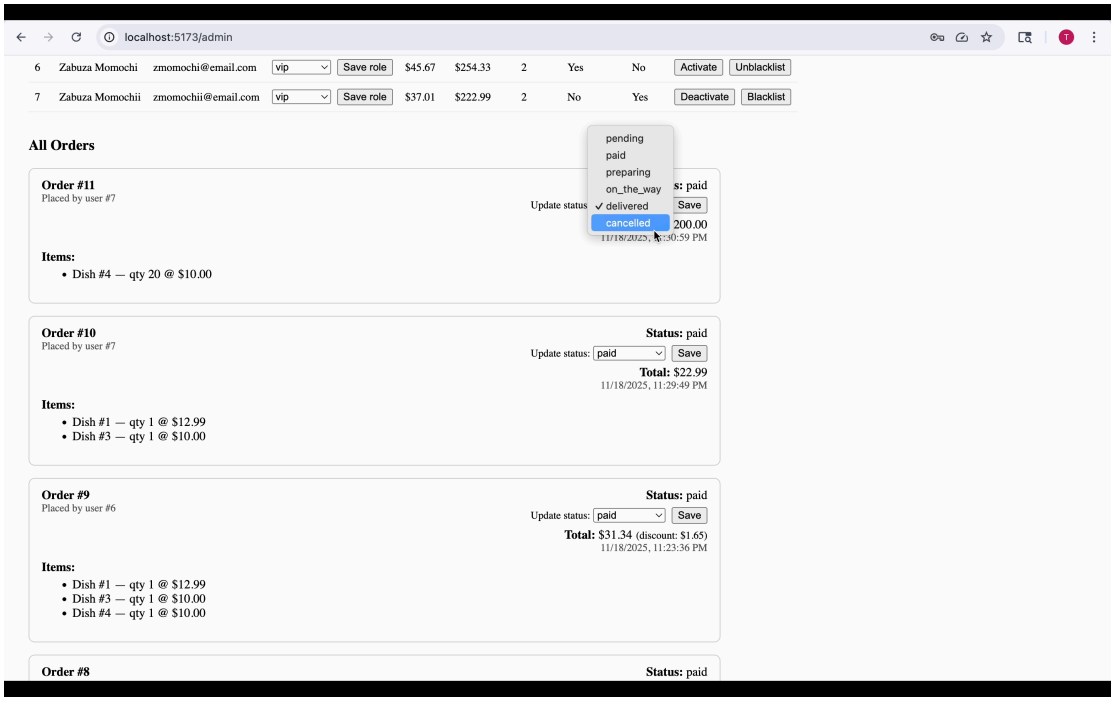
Main Features:

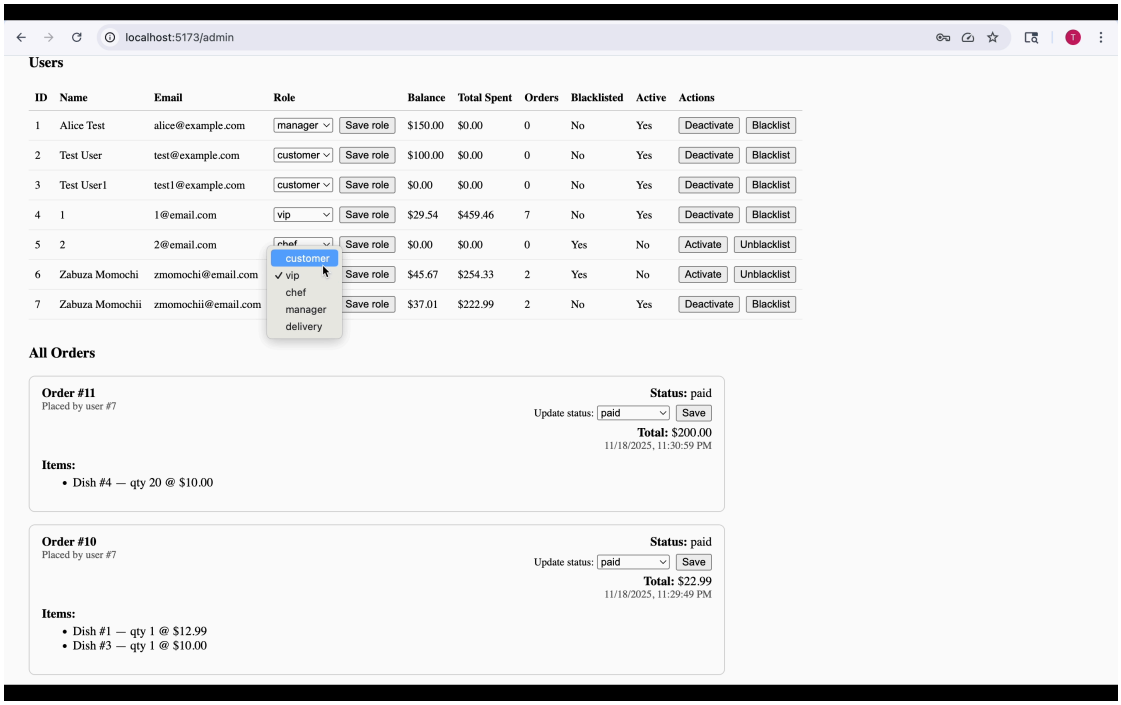
- Links to admin functions: **Manage Menu**, **Manage Orders**, **View Reports**, etc.
- Menu list with options to **Add**, **Edit**, or **Deactivate** items.
- Order queue for processing and delivery assignment.




localhost:5173/admin
ID Name Email Role Balance Total Spent Orders Blacklisted Active Actions
1 Alice Test alice@example.com manager Save role $150.00 $0.00 0 No Yes Deactivate Blacklist
2 Test User test@example.com customer Save role $100.00 $0.00 0 No Yes Deactivate Blacklist
3 Test User1 test1@example.com customer Save role $0.00 $0.00 0 No Yes Deactivate Blacklist
4 1 1@email.com vip Save role $29.54 $459.46 7 No Yes Deactivate Blacklist
5 2 2@email.com chef Save role $0.00 $0.00 0 Yes No Activate Unblacklist
6 Zabuza Momochi zmomochi@email.com vip Save role $45.67 $254.33 2 Yes Yes Deactivate Unblacklist
7 Zabuza Momochii zmomochii@email.com vip Save role $37.01 $222.99 2 No Yes Deactivate Blacklist
All Orders
Order #11
Placed by user #7
Status: paid
Update status: paid Save
Total: $200.00
11/18/2025, 11:30:59 PM
Items:
• Dish #4 — qty 20 @ $10.00
Order #10
Placed by user #7
Status: paid
Update status: paid Save


localhost:5173/admin
6 Zabuza Momochi zmomochi@email.com vip Save role $45.67 $254.33 2 Yes No Activate Unblacklist
7 Zabuza Momochii zmomochii@email.com vip Save role $37.01 $222.99 2 No Yes Deactivate Blacklist
All Orders
Order #11
Placed by user #7
Update status paid Save
pending
paid
preparing
on_the_way
delivered
cancelled
Items:
• Dish #4 — qty 20 @ $10.00
Order #10
Placed by user #7
Status: paid
Update status: paid Save
Total: $22.99
11/18/2025, 11:29:49 PM
Items:
• Dish #1 — qty 1 @ $12.99
• Dish #3 — qty 1 @ $10.00
Order #9
Placed by user #6
Status: paid
Update status: paid Save
Total: $31.34 (discount: $1.65)
11/18/2025, 11:23:36 PM
Items:
• Dish #1 — qty 1 @ $12.99
• Dish #3 — qty 1 @ $10.00
• Dish #4 — qty 1 @ $10.00
Order #8
Status: paid





10. Delivery Page

Purpose:

The Delivery Page allows users (or delivery staff/admin) to view

the delivery status of an active order. It centralizes delivery tracking information and provides real-time updates as the order moves through the delivery pipeline.

Main Features:

- Shows **current delivery status** (e.g., *Preparing*, *Out for Delivery*, *On the Way*, *Delivered*).
- Displays **estimated delivery time** (if implemented).
- Displays **delivery address** and contact details.
- May include a **map view** or **location updates** (optional depending on implementation).
- Shows name of the **assigned delivery driver** (if the system includes that component).
- Status updates appear automatically or on page refresh.

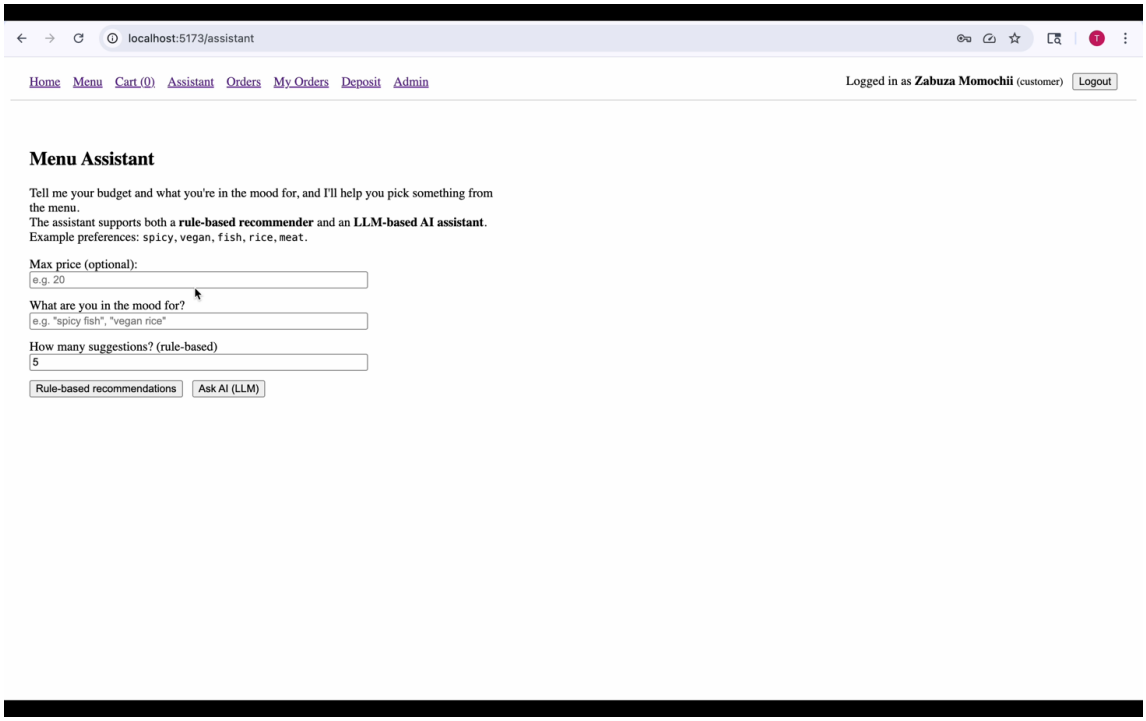
Functionality of your own choice

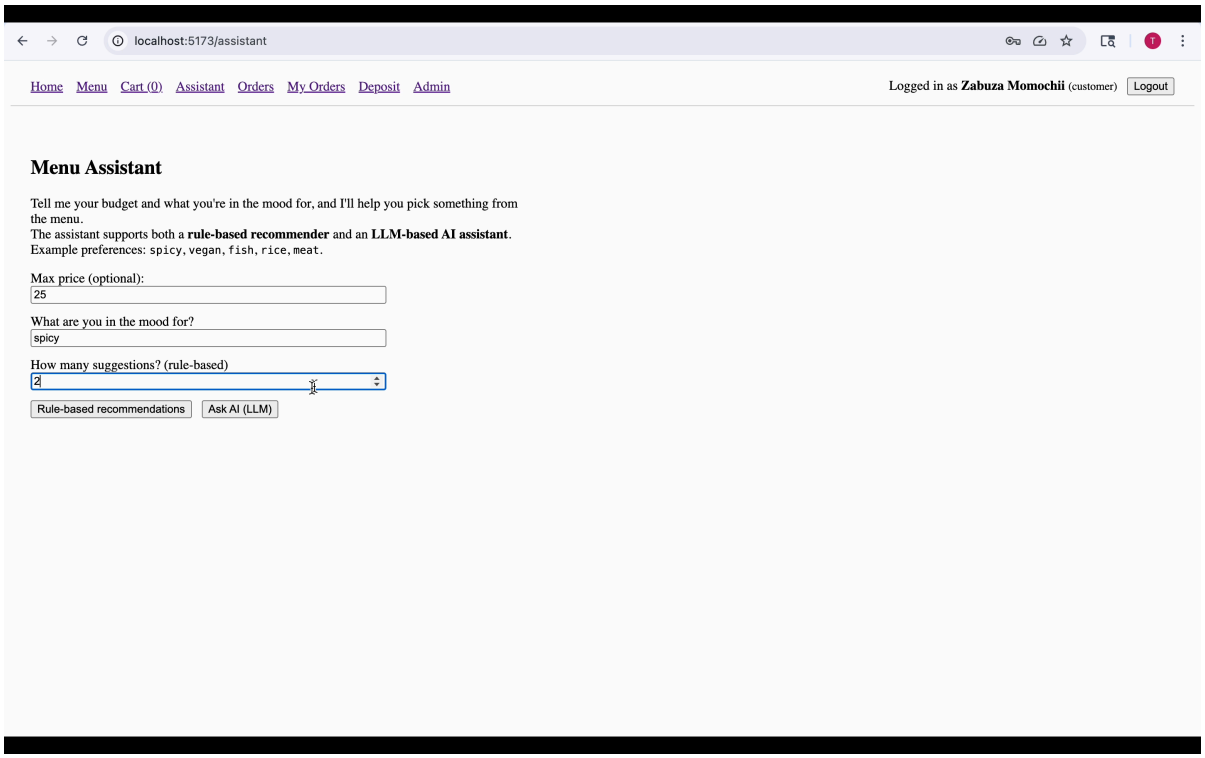
Purpose:

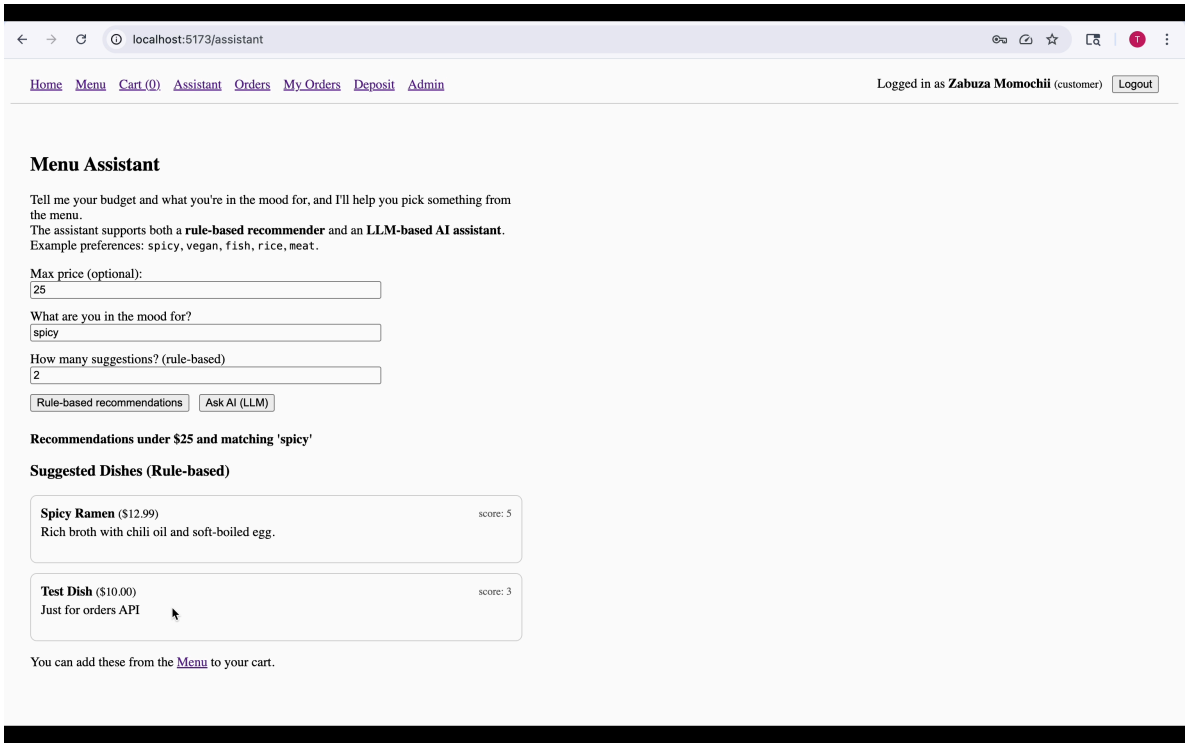
Provides AI or help functionality for the user (optional feature in your app).

Main Features:

- Input field to ask questions.
- AI-generated responses or guidance.


localhost:5173/assistant
Home Menu Cart (0) Assistant Orders My Orders Deposit Admin
Logged in as Zabuza Momochii (customer) Logout
Menu Assistant
Tell me your budget and what you're in the mood for, and I'll help you pick something from the menu.
The assistant supports both a rule-based recommender and an LLM-based AI assistant.
Example preferences: spicy, vegan, fish, rice, meat.
Max price (optional):
e.g. 20
What are you in the mood for?
e.g. "spicy fish", "vegan rice"
How many suggestions? (rule-based)
5
Rule-based recommendations
Ask AI (LLM)


localhost:5173/assistant
Home Menu Cart (0) Assistant Orders My Orders Deposit Admin
Logged in as Zabuza Momochii (customer) Logout
Menu Assistant
Tell me your budget and what you're in the mood for, and I'll help you pick something from the menu.
The assistant supports both a rule-based recommender and an LLM-based AI assistant.
Example preferences: spicy, vegan, fish, rice, meat.
Max price (optional):
25
What are you in the mood for?
spicy
How many suggestions? (rule-based)
2
Rule-based recommendations
Ask AI (LLM)



Division of work among group members:

Amir Nabiyev – Introduction/ Use Case collaboration diagrams

Mohamed Komagate – Use Case Collaboration diagrams and the ER diagram

Oumar Kante – GUI and feature prototypes

Tanzila Rehman – Pseudo Code Functionality

GitHub repo:

https://github.com/AmirNabiyev30/Cs_322_AI_Enabled_Resturant_Group_F