Project Name - Pizza

Intro to Software Engineering Section 01

Ahmadou Bah,

William Donovan,

Jaymin McCoy,

Wainaina Wainaina,

Panic Party Pizzeria

Functional Requirements

FR1) Customer Management

1.1) The system shall allow new customers to sign up by entering their name, phone number, address, and preferred card type.
1.2) The system shall allow returning customers to log in using their phone number.
1.3) The system shall store customer records (name, phone number, address, card type, and payment preferences) in the database.
1.4) The system shall allow customers to view and update their personal information (name, address, and card type).
1.5) The system shall classify users as “New” or “Returning” based on signup/login activity.

FR2) Menu Management

2.1) The system shall display a menu UI that includes pizza options (size, crust type, toppings) and beverages.
2.2) The system shall allow customers to select items using checkboxes or toggles.
2.3) The system shall display item prices and automatically calculate the total cost of an order.
2.4) The system shall paginate menu items if the list is large.

FR3) Order Processing

3.1) The system shall record each order placed by a customer.
3.2) The system shall store order details such as selected items, quantities, total cost, and payment type.
3.3) The system shall display an order summary screen showing itemized costs and total price.
3.4) The system shall support “Pay at Store” as an option for carryout orders.
3.5) The system shall display a “Thank You” screen with order confirmation and estimated pickup/delivery time.
3.6) The system shall generate a digital receipt for each order.
3.7) The system shall store receipts in the database for future reference.
3.8) The system shall include a signature field for card transactions.

FR4) Payment Processing

4.1) The system shall support both card and cash payments.
4.2) The system shall securely record the payment method used (cash, card, or pay at store).

4.3) The system shall validate and process card payments via an integrated payment gateway.

FR5) Employee and Manager Operations

5.1) The system shall allow employees and managers to log in via phone number authentication.
5.2) The system shall provide clock-in/out functionality for employees.
5.3) The system shall store and track employee information (e.g., names, shifts).
5.4) The system shall allow managers to:
5.5) View and update inventory.
5.6) Access all previous orders and receipts.
5.7) View and analyze sales data by day, week, and month.
5.8) Perform refunds.
5.9) View total profit calculations.

FR6) Authentication and Authorization

6.1) The system shall authenticate users via phone number with code verification (2FA).
6.2) The system shall restrict access based on user roles:
6.3) Customers can access ordering and payment functions.
6.4) Employees can clock in/out and process orders.
6.5) Managers can access all administrative features.
6.6) The system shall limit the number of accounts to 3 employees and 1 manager.

FR7) UI & Navigation

7.1) The system shall have the following screens:
7.2) Opening Screen – shows business name, address, phone number, and login options.
7.3) Login/Signup Screen – user authentication by phone number.
7.4) Menu Selection Screen – interactive menu of pizzas and beverages.
7.5) Order Summary/Payment Screen – display of order total and payment options.
7.6) Thank You Screen – order confirmation.
7.7) Manager Dashboard – access to reports, inventory, and employee data.
7.8) The UI shall be responsive and work across smartphones, tablets, and desktops.

FR8) Reporting & Analytics

8.1) The system shall allow managers to view sales reports by day, week, and month.
8.2) The system shall calculate and display total profits.
8.3) The system shall log and store all refund and transaction activities for reporting.

FR9) Data Storage & Management

9.1) The system shall store all customer, order, and payment data in a PostgreSQL database.
9.2) The system shall encrypt sensitive information (e.g., payment details).
9.3) The system shall implement pagination for order history and menu data.
9.4) The system shall log all manager actions for auditing purposes.

Nonfunctional Requirements

NFR1) Responsiveness

1.1) User interface shall be responsive and provide a consistent, functional experience across all target devices
1.2) The system will be used by customers on their phones

NFR2) Clarity

2.1) All interactive elements (buttons, checkboxes, toggles) must clearly indicate their state (e.g., selected, unselected)
2.2) Users need to be confident in their selections. When a customer adds toppings to a pizza, each selected topping must be visually distinct from unselected ones. This prevents confusion and errors in the order

NFR3) Feedback

3.1) The system shall provide clear and immediate feedback for user actions, such as successful item addition, login failure, or successful payment
3.2) The system must communicate with the user. If a user clicks "Add to Order" and nothing happens, they will click it again, resulting in duplicate items. If a login fails, the user must be told why

NFR4) Speed

4.1) All primary screens (Menu, Order Summary, Manager Dashboard) shall load completely within 3 seconds
4.2) Slow-loading pages are the primary reason users abandon an application. The 3-second mark is an industry standard for retaining user engagement. The Manager Dashboard, in particular, must be fast so a manager can get sales data quickly

NFR5) Proper Payment

5.1) Payment authorization (card processing) must return a success or failure status to the user within 5 seconds

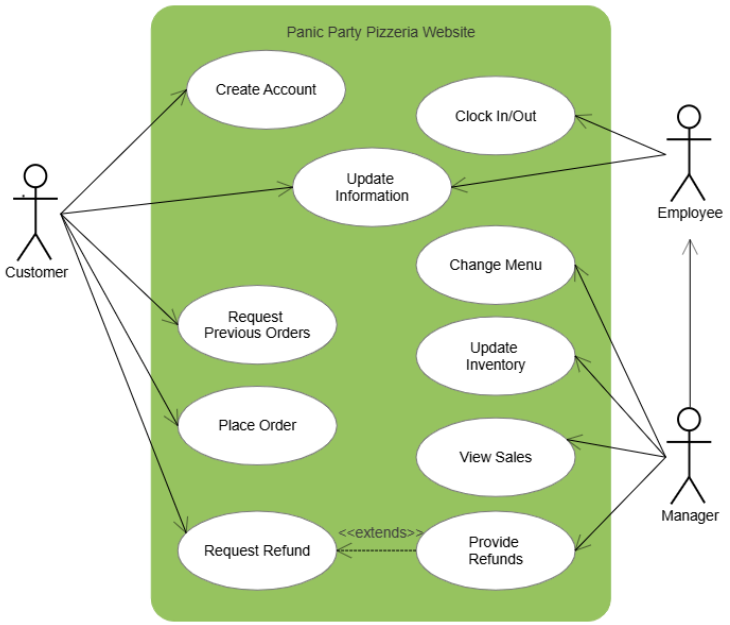
5.2) This is the most sensitive part of the user flow. If a payment hangs for more than a few seconds, the user will get nervous, assume it failed, and may try to pay again or abandon the order

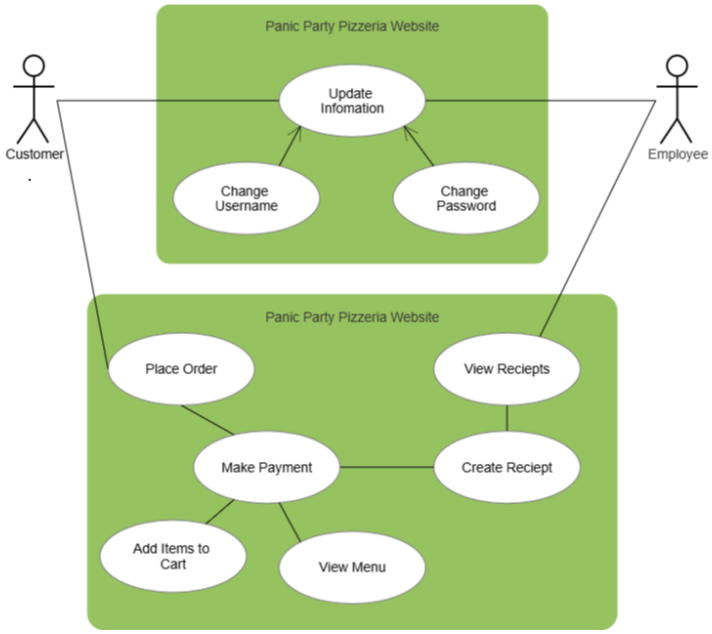
NFR6) Security

6.1) Employee-level accounts must be prohibited from accessing manager-level functions

6.2) This is a core business and security rule. An employee should be able to place an order but never see sales reports, perform refunds, or view other employee information.

Use Case Diagram





Case Flow

1.0 Flow of events for the "Place Order" use case

1.1 Preconditions

- The customer must be logged into the website.
- The customer has selected items from the menu and added them to their cart.

1.2 Main Flo

This use case begins when the customer navigates to the "Place Order" page. The system displays the cart with the selected items.

- The customer reviews the items in the cart and confirms they are ready to place the order.
- The system prompts the customer to proceed with payment.
- The customer enters payment details.
- The system processes the payment information.
- If the payment is successful, the system confirms the order and provides the order details along with an estimated delivery time.
- The system ends the process by displaying the order confirmation.

1.3 Subflows

- **S-1: Add Items to Cart**
    - The customer selects items from the menu.
    - The system adds the items to the cart.
    - The customer can modify the quantity or remove items.
    - The system updates the cart accordingly.
- **S-2: Make Payment**
    - The customer enters payment information.
    - The system verifies the payment details.
    - The payment is processed.
    - The system confirms the payment and the order is placed.

1.4 Alternative Flows

- **E-1: Payment Fails**
    - **E-1.1** If the payment fails, the system notifies the customer that payment could not be processed.
    - **E-1.2** The customer is given an option to re-enter payment details or use an alternative payment method.
- **E-2: Insufficient Stock**
    - **E-2.1** If any item in the cart is out of stock, the system informs the customer.
    - **E-2.2** The customer is prompted to modify their cart or choose alternative items.

2.0 Flow of events for the "Request Refund" use case

2.1 Preconditions

- The customer must be logged into the website.
- The customer must have a past order to request a refund for.

2.2 Main Flow

This use case begins when the customer navigates to the "View Receipts" page.

- The system displays a list of the customer's previous orders.
- The customer selects the order for which they want a refund.
- The system asks the customer to provide a reason for the refund request.
- The customer enters the reason for the refund.
- The system processes the refund request and updates the order status.
- The system confirms that the refund request is being processed and sends a notification to the customer.

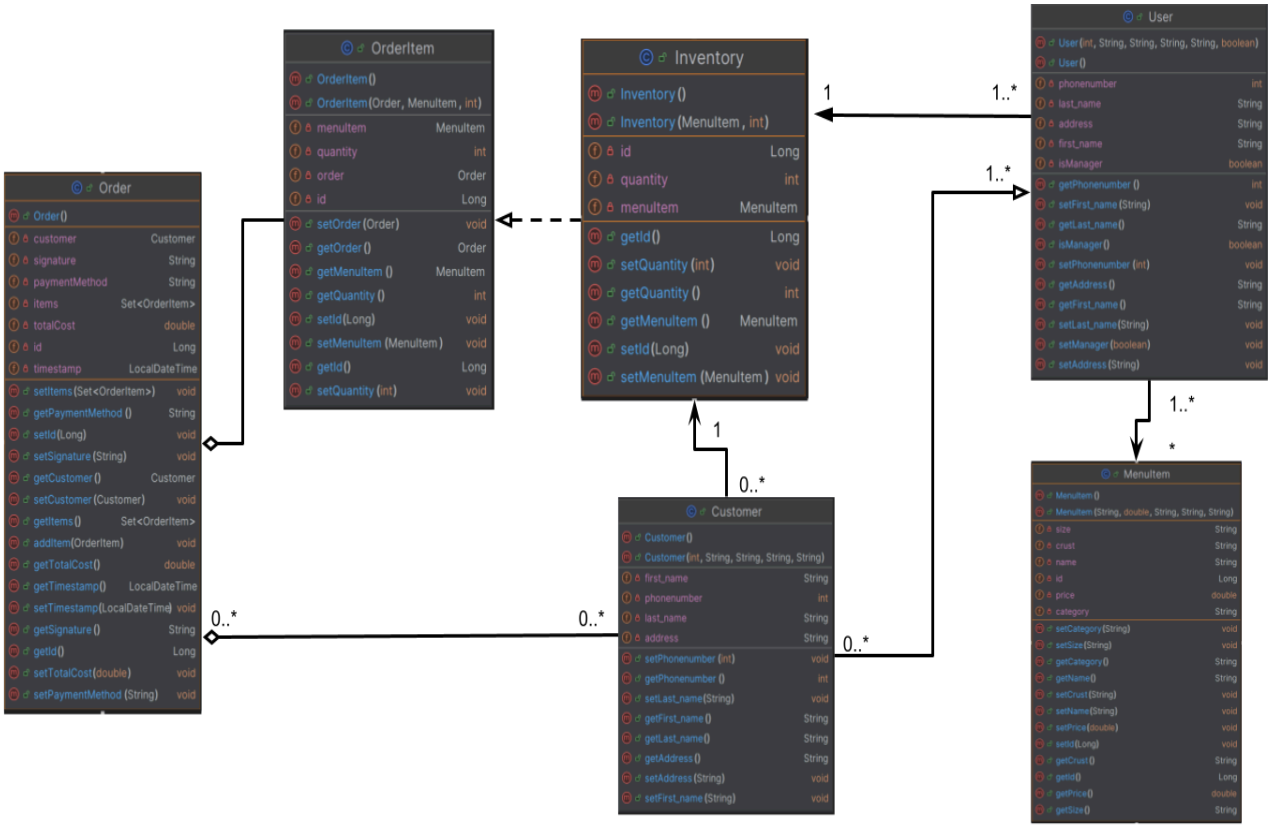
2.3 Subflows

- **S-1: View Receipts**
    - The customer selects the "View Receipts" option from the menu.
    - The system retrieves and displays all previous orders.
    - The customer selects an order to view more details.
- **S-2: Process Refund**
    - The system prompts the customer to enter the reason for the refund.
    - The customer enters the reason.
    - The system processes the refund request and updates the order status.

2.4 Alternative Flows

- **E-1: Invalid Order**
    - **E-1.1** If the selected order is invalid (e.g., no longer eligible for a refund), the system informs the customer that the order cannot be refunded.
    - **E-1.2** The customer is prompted to select another order or terminate the refund process.
- **E-2: Refund Criteria Not Met**
    - **E-2.1** If the refund does not meet the refund policy criteria (e.g., request made after the allowed time), the system denies the refund request.
    - **E-2.2** The system informs the customer that the refund cannot be processed.

UML Diagram



Class Documentation

Project Purpose

This document outlines the technical design for a modern Point-of-Sale (POS) system for a restaurant. The system is a full-stack web application designed to manage customer orders, staff logins, menu items, and sales reporting.

The frontend will be a React.js application, allowing customers and staff to access the system from any device (phone, tablet, or desktop). The backend will be a Java Spring Boot API responsible for all business logic.

Technology Stack

Frontend: React.js

Backend: Java + Spring Boot

Database: PostgreSQL

System Architecture

1. Controller (*Controller.java)

Purpose: It handles incoming HTTP requests

Job: It validates input, calls the Service layer to do the work, and returns an HTTP response.

2. Service (*Service.java)

Purpose: Handles business logic.

Job: It contains all business logic (e.g., "check if a customer exists," "calculate order total," "check if user is a manager"). It calls the Repository to get data.

3. Repository (*Repository.java)

Purpose: It's an interface that talks directly to the database.

Job: It performs all CRUD (Create, Read, Update, Delete) operations, like findAll(), findById(), and save().

4. Entity (*.java)

Purpose: Serve as a model of the data in the database.

Job: A simple Java class that directly maps to a table in the PostgreSQL database.

User (Staff)

Represents a staff member who logs into the system. This class handles permissions and staff-related actions.

Customer

Represents a paying customer. This class stores their personal info for placing orders and viewing order history.

MenuItem

Represents a template for a single item on the restaurant's menu.

Inventory

Tracks the stock level for a single menu item.

Order

Represents a single transaction.

OrderItem

This is a single line item on the receipt. It connects an Order to a MenuItem and stores the quantity.

Key Design Decisions

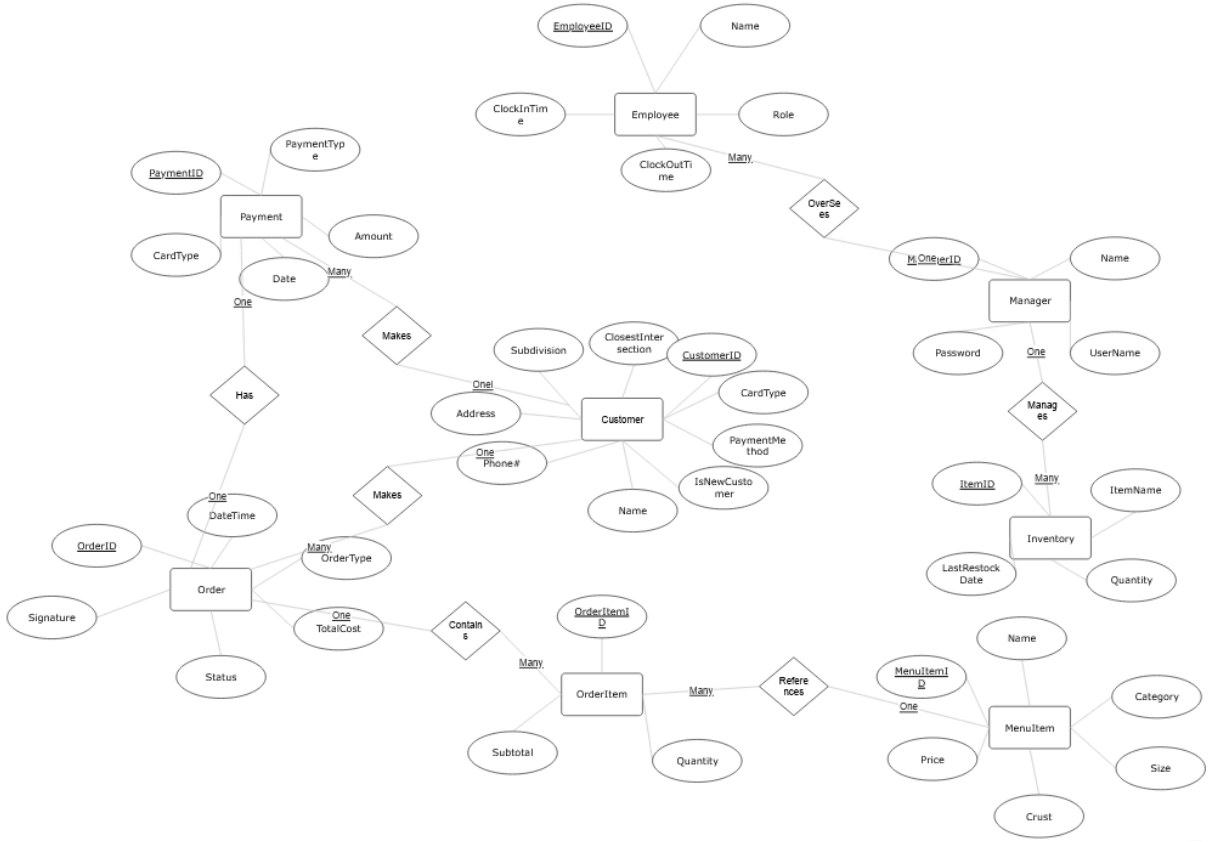
Staff vs. Customers: Staff (User) and Customer are two separate classes. This is intentional. A User has a password and a role for security. A Customer has an address and cardType for placing orders.

Role-Based Access (User Class). One class for employees with fields differentiating manager from the employee.

Order vs. OrderItem: We must have both classes. MenuItem is the template. OrderItem is the instance on a receipt. This "bridge" class is the standard way to model a many-to-many relationship.

String for Phone Number: All phoneNumber fields are String, not int. This is critical to store leading zeros, international codes (e.g., +1), and formatting characters like (,), and -.

ER Diagram



Decision Tables

Sales Logic	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8	Actions
Existing Customer?	T	T	T	T	F	F	F	F	
Payment Type = Credit Card?	T	T	F	F	T	T	F	F	
Payment Approved?	T	F	–	–	T	F	–	–	
Order Type = Delivery?	T	T	T	F	T	T	T	F	
	X	X	X	X	–	–	–	–	Retrieve customer record
	–	–	–	–	X	X	X	X	Create new customer record
	X	–	–	–	X	–	–	–	Process credit card payment
	–	–	X	X	–	–	X	X	Accept cash payment
	X	X	X	X	X	X	X	X	Print receipt (include signature line if credit)
	X	X	X	–	X	X	X	–	Assign delivery driver
	X	–	X	X	X	–	X	X	Mark order as complete
	–	X	–	–	–	X	–	–	Display “Payment Declined” message

Signup	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8	Rule 9
Phone Number Status	Taken	Taken	Taken	Invalid	Invalid	Invalid	Valid	Valid	Valid
Password Length	6>	15<	Valid	6>	15<	Valid	6>	15<	Valid
Result	Taken Error	Taken Error	Taken Error	Invalid Number Error	Invalid Number Error	Invalid Number Error	Less than 6	More than 15	Allow Signup

Login	Rule 1	Rule 2	Rule 3	Rule 4
Is Number Correct?	T	T	F	F
Is Password Correct?	T	F	T	F
Result	Login	Incorrect Information Error	Incorrect Information Error	Incorrect Information Error

Cart Actions	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
Is Cart Empty?	Empty	Not Empty	Full	Empty	Not Empty	Full
Is Item Available?	T	T	T	F	F	F
Result	Add to Cart	Add to Cart	Full Cart Error	Unavailable Error	Unavailable Error	Unavailable Error

Orders	Rule 1	Rule 2	Rule 3	Rule 4
Is Store Open?	T	T	F	F
Is Cart Empty?	Empty	Not Empty	Empty	Not Empty
Result	Empty Cart Error	Place Order	Store Closed Error	Store Closed Error