Project : 1c1

Project Group 6 - Section 1

Project Members:

Name	Email
Khush Ketan Patel	kpatel53@ncsu.edu
Mia Glenn	mglenn2@ncsu.edu
Richa Jha	rjha3@ncsu.edu
Ishwarya Gandamsetty	igandam@ncsu.edu

New Use Cases:

Use Case: Collaborative Group Ordering

Description

Multiple users add items to a shared order before submission.

Preconditions

The initiator is logged in and has created a group order session.

Main Flow:

1. The initiator creates a group order and sets a cutoff time.
2. The initiator invites participants [Invite Participants].
3. Participants open the link and join the session [Join Session].
4. Participants browse the selected restaurant and add items to the shared cart [Add Items].
5. The initiator reviews the cart and resolves conflicts [Resolve Conflicts].
6. The initiator finalizes the group order at or before cutoff [Finalize Group Order].

Subflows

- [Invite Participants] The initiator shares a join link via in-app contacts, SMS, or email. The system tracks invite status.
- [Join Session] The participant accepts the invite, provides a display name if needed, and enters the session.
- [Add Items] A participant adds items to the shared cart. The system locks the item row during editing to prevent collisions. On save, the cart updates for all members.
- [Resolve Conflicts] The initiator reviews duplicate items or exceeded per-item limits. The initiator can merge duplicates, remove items, or request changes [Request Changes].
- [Request Changes] The initiator pings specific participants with comments. Participants adjust or confirm their items.
- [Finalize Group Order] The system locks the cart, calculates prep time, and places the order. The use case ends.

Alternative Flows:

- [No Response] Invited participants do not join before cutoff. Their seats are released and the order proceeds without them.
- [Restaurant Unavailable] The restaurant disables orders during the session. The initiator selects a replacement restaurant or the session ends without placement.
- [Participant Limit Reached] Additional join attempts are rejected with a message.

Use Case: Curbside Pickup Handoff

Description:

Coordinate restaurant staff and customer for curbside pickup outside the venue.

Preconditions

The customer selected curbside pickup and provided vehicle details. The restaurant accepted the order.

Main Flow:

1. The customer drives to the restaurant location.

2. The customer announces arrival via the app [Announce Arrival].
3. The system assigns a staff runner and pickup slot [Assign Runner].
4. The runner locates the vehicle and verifies identity [Verify Vehicle].
5. The runner hands off the order [Handoff] and the use case ends.

Subflows:

- [Announce Arrival] The customer taps “I’m here,” confirming parking spot number or geofenced arrival; the system shares vehicle make, model, color, and spot.
- [Assign Runner] The kitchen marks the order “ready.” The system notifies floor staff and selects a runner based on proximity and workload.
- [Verify Vehicle] The runner matches plate, color, and order name. If mismatched, the runner contacts the customer via in-app call [Contact Customer].
- [Handoff] The runner confirms the order code with the customer, places the package in the trunk or back seat per customer preference, and captures a confirmation photo.
- [Contact Customer] The runner calls or messages the customer to confirm location and identity, then returns to [Verify Vehicle].

Alternative Flows:

- [Customer Not Arrived] The customer has not arrived within the grace period. The order is moved to holding shelves and pickup switches to in-store counter.
- [Parking Unavailable] Lot is full. The app instructs the customer to use a designated loading zone and updates staff.
- [Safety Concern] Weather or visibility issues prevent safe curbside handoff. The app switches to doorway pickup with instructions.

Use Case: Tamper-Evident Packaging Verification

Description:

Ensure orders are sealed and tamper-free at pickup and at delivery.

Preconditions:

The restaurant uses tamper-evident seals. The driver app supports photo capture.

Main Flow:

1. The restaurant packages the order and applies seals [Seal Order].
2. The driver arrives for pickup and inspects seals [Inspect at Pickup].
3. The driver transports the order.
4. The driver presents the order to the customer and captures delivery confirmation [Inspect at Delivery].

Subflows:

- [Seal Order] Staff closes all containers and bags, applies numbered or QR-coded seals, and marks “sealed” in the order system.
- [Inspect at Pickup] The driver checks that all containers are sealed and codes are intact. If an issue is found, the driver requests reseal [Request Reseal] before leaving.
- [Request Reseal] The restaurant re-packages or re-seals the order, updates seal codes, and confirms readiness.
- [Inspect at Delivery] The driver shows intact seals to the customer, captures a photo of seals with codes, and the use case ends.

Alternative Flows:

- [Seal Missing] No seal is present. The driver requests sealing; if the restaurant cannot comply, the driver flags the order and may decline pickup per policy.
- [Seal Broken in Transit] A seal appears damaged at delivery. The driver informs the customer and support is notified to determine remediation.
- [Mismatch Codes] The codes in the app do not match physical seals. The driver requests verification; if unresolved, proceed to [Seal Broken in Transit].

Use Case: Secure Building Access Code Exchange

Description:

Safely obtain and use temporary access credentials to deliver to secure buildings.

Preconditions:

The delivery address requires restricted entry. The platform supports temporary credential exchange.

Main Flow:

1. The customer provides access instructions during checkout [Provide Access].
2. The system requests or generates a time-bound access credential [Generate Credential].
3. The driver approaches the building and retrieves the credential [Retrieve Credential].
4. The driver gains entry and completes delivery [Use Credential].

Subflows:

- [Provide Access] The customer selects access method (temporary code, intercom, concierge) and optionally sets a time window.
- [Generate Credential] For supported buildings, the system requests a one-time PIN or QR from the building API; otherwise stores customer-provided instructions encrypted.
- [Retrieve Credential] Upon arrival, the driver’s app verifies proximity and reveals the credential. Credentials remain hidden until within geofence.
- [Use Credential] The driver enters the code or presents the QR at the door panel, reaches the unit or mailroom, delivers per instructions, and the use case ends.

Alternative Flows:

- [Credential Expired] The code has expired. The driver requests a fresh code via the app; if unavailable, the system switches to intercom or concierge workflow.
- [No Connectivity] The driver cannot fetch the credential due to poor signal. The app shows offline instructions or a fallback PIN cached during dispatch.

- [Access Denied] The building rejects the credential. The driver contacts the customer and support; if access cannot be established within the wait window, the order is returned per policy.

Use Case: AI-Powered Supplier Forecasting for Restaurants

Description:

Safely obtain and use temporary access credentials to deliver to secure buildings.

Preconditions:

Restaurant inventory data integrated.

Main Flow:

1. System analyzes order trends [Analyze Trends].
2. Forecast ingredient demand [Forecast Demand].
3. Recommend supplier orders [Recommend Orders].

Subflows:

- [Analyze Trends] Look at daily/weekly order patterns.
- [Forecast Demand] Predict ingredient needs.

Alternative Flows:

- [Unexpected Event] Adjust forecast based on holidays/events.
- [Supplier Shortage] Recommend alternate suppliers.

Use Case: Nutritional Gamification for Customers

Description:

Customer has enabled health goals.

Preconditions:

Restaurant inventory data integrated.

Main Flow:

1. System tracks ordered meals [Track Meals].
2. Award points for healthy selections [Award Points].
3. Display streaks and achievements [Display Rewards].

Subflows:

- [Track Meals] Use nutrition engine to validate “healthy” meals.
- [Award Points] Apply gamified logic (badges, levels).

Alternative Flows:

- [Unhealthy Streak] Suggest healthier alternatives.
- [Customer Opts Out] Disable gamification.

Use Case: AI Prediction of Delivery Delays

Description:

Orders in transit.

Preconditions:

Restaurant inventory data integrated.

Main Flow:

1. System monitors driver GPS [Track Driver].
2. AI predicts delay likelihood [Predict Delay].
3. Customer notified proactively [Notify Customer].

Subflows:

- [Predict Delay] Use traffic, weather, and driver history.
- [Notify Customer] Provide revised ETA.

Alternative Flows:

- [False Alarm] Delivery on time → auto-correct ETA.
- [Severe Delay] Offer coupon/discount.

Use Case: AI Chatbot for Restaurant Staff Training

Description:

Orders in transit.

Preconditions:

Restaurant has enabled training module.

Main Flow:

1. New staff logs into training [Access Training].
2. AI chatbot simulates customer interactions [Simulate Interaction].
3. Staff completes training session [Complete Training].

Subflows:

- [Simulate Interaction] Generate real-world order scenarios.
- [Complete Training] Track performance and feedback.

Alternative Flows:

- [Staff Fails Assessment] Recommend re-training.
- [System Error] Switch to manual training.

Use Case: Driver Registration and Account Verification

Description:

New delivery drivers complete the onboarding process by registering their account, uploading required documentation, and undergoing background verification to become authorized delivery personnel.

Preconditions:

- Driver has valid government-issued identification and driver's license
- Driver owns or has access to approved delivery vehicle
- Driver has smartphone with GPS capability and camera
- System has active integration with background check services

Main Flow:

1. Driver initiates registration through mobile app or web portal

2. Driver provides personal information and contact details [Personal Information Entry]
3. Driver uploads required documentation and vehicle information [Document Upload]
4. System initiates automated background and driving record checks [Verification Process]
5. Driver completes vehicle inspection and insurance verification [Vehicle Certification]
6. System approves driver account and grants delivery access permissions [Account Activation]

Subflows:

- [Personal Information Entry] Driver enters full name, address, phone number, emergency contact, and banking details for payments
- [Document Upload] Driver photographs and uploads driver's license, vehicle registration, insurance certificate, and profile photo
- [Verification Process] System validates documents against government databases and initiates third-party background screening
- [Vehicle Certification] Driver submits vehicle photos, passes safety checklist, and confirms insurance coverage meets minimum requirements

Alternative Flows:

- [Document Rejection] If uploaded documents are unclear or expired, system requests re-submission with specific guidance on requirements
- [Background Check Failure] If background check reveals disqualifying issues, system denies application and provides appeal process information
- [Incomplete Application] If driver abandons registration mid-process, system saves progress and sends reminder notifications to complete within 30 days

Use Case: Restaurant Performance Analytics Dashboard

Description:

Restaurant managers access comprehensive analytics to monitor operational performance, identify trends, and make data-driven decisions to optimize their business operations.

Preconditions:

- Restaurant manager has authenticated administrative access
- Restaurant has been operational for minimum 30 days with order history
- System has collected sufficient data across multiple operational metrics
- Analytics engine has processed recent transaction and performance data

Main Flow:

1. Manager logs into restaurant analytics portal during business review period
2. Manager selects reporting timeframe and performance categories [Report Configuration]
3. System generates comprehensive performance metrics and visualizations [Data Analysis]
4. Manager reviews key performance indicators and trend analysis [Performance Review]
5. Manager identifies areas for improvement and exports actionable insights [Decision Making]

Subflows:

- [Report Configuration] Manager selects date ranges, compares periods, and customizes dashboard widgets for specific metrics
- [Data Analysis] System calculates order volumes, average preparation times, customer ratings, revenue trends, and peak hour patterns
- [Performance Review] Manager examines customer satisfaction scores, delivery success rates, menu item popularity, and operational efficiency metrics
- [Decision Making] Manager generates reports for staff meetings, identifies underperforming items, and plans operational adjustments

Alternative Flows:

[Insufficient Data] If selected timeframe lacks adequate data, system suggests alternative periods and provides data collection recommendations

[System Maintenance] If analytics engine is updating, system displays cached reports with timestamp indicating last refresh

[Comparative Analysis Request] If manager wants to benchmark against similar restaurants, system provides anonymized industry comparisons where available

Use Case: Premium Membership Enrollment and Benefits Management

Description:

Customers enroll in premium membership programs to access exclusive benefits, priority service, and special rewards while the system manages their membership status and benefit redemption.

Preconditions:

- Customer has active account with verified payment method
- Customer has completed minimum number of orders to qualify for membership
- System has active membership tier structure and benefit catalog
- Payment processing system is operational for subscription handling

Main Flow:

1. Customer views membership options and benefit details in app
2. Customer selects desired membership tier and reviews terms [Membership Selection]
3. Customer completes enrollment and payment setup for recurring subscription [Enrollment Process]
4. System activates membership benefits and updates customer profile [Benefit Activation]
5. Customer begins utilizing premium features and exclusive offers [Benefit Utilization]

Subflows:

- [Membership Selection] Customer compares tier benefits including free delivery thresholds, exclusive menu access, and priority support
- [Enrollment Process] Customer confirms subscription terms, sets up automatic payments, and receives membership confirmation
- [Benefit Activation] System unlocks premium features, applies delivery fee waivers, and enables access to member-only restaurants
- [Benefit Utilization] Customer places orders with automatic benefit application and tracks membership savings and rewards

Alternative Flows:

- [Payment Failure] If subscription payment fails, system provides grace period and multiple retry attempts before suspending benefits
- [Membership Cancellation] If customer cancels membership, system maintains benefits through current billing cycle and offers retention incentives
- [Benefit Dispute] If customer reports benefit not applied correctly, system reviews order history and provides automatic credit or manual resolution

Use Case: Fraudulent Order Detection and Prevention

Description:

The system automatically monitors order patterns and payment activities to detect suspicious behavior, prevent fraudulent transactions, and protect both customers and restaurants from financial losses.

Preconditions:

- System has established baseline patterns for normal customer behavior
- Fraud detection algorithms are trained with historical transaction data
- Integration with payment processor fraud monitoring is active
- Customer support team is available for fraud investigation escalation

Main Flow:

1. System continuously monitors incoming orders for suspicious patterns [Pattern Analysis]

2. System identifies potentially fraudulent activity based on risk indicators [Risk Assessment]
3. System applies appropriate security measures based on fraud probability [Security Response]
4. System notifies relevant parties and initiates verification procedures [Verification Process]
5. System either approves legitimate orders or blocks fraudulent attempts [Final Decision]

Subflows:

- [Pattern Analysis] System analyzes order frequency, payment methods, delivery addresses, and device fingerprints against established behavioral models
- [Risk Assessment] System calculates fraud probability using factors like new account activity, unusual order values, and geographic inconsistencies
- [Security Response] System may require additional authentication, hold orders for manual review, or temporarily restrict account privileges
- [Verification Process] System sends verification requests to customers and alerts restaurant partners about potentially problematic orders

Alternative Flows:

- [False Positive Detection] If legitimate customer is flagged incorrectly, system provides expedited appeal process and adjusts detection algorithms
- [Confirmed Fraud Case] If fraud is confirmed, system permanently blocks offending accounts, refunds affected parties, and updates prevention models
- [Payment Processor Alert] If external payment system flags transaction, system coordinates response and may require alternative payment verification

Use Case: Multi-Restaurant Order Coordination and Fulfillment

Description:

Customers place orders from multiple restaurants simultaneously, and the system coordinates preparation timing, delivery logistics, and payment processing to ensure synchronized delivery of the complete order.

Preconditions:

- Customer has items from multiple restaurants in shopping cart
- All selected restaurants are operational and accepting orders
- Delivery driver capacity is available for multi-restaurant pickup route
- System can calculate optimal timing coordination between restaurants

Main Flow:

1. Customer finalizes multi-restaurant order with items from different establishments [Order Confirmation]
2. System calculates optimal preparation timing and delivery coordination [Logistics Planning]
3. System distributes order components to respective restaurants with timing instructions [Order Distribution]
4. System assigns specialized driver and creates optimized pickup route [Driver Assignment]
5. System coordinates real-time updates and ensures synchronized delivery [Delivery Coordination]

Subflows:

- [Order Confirmation] System validates item availability across all restaurants, calculates total pricing including coordination fees, and confirms delivery address
- [Logistics Planning] System determines preparation time differences, establishes pickup sequence, and sets target delivery window
- [Order Distribution] System sends timed order notifications to each restaurant with specific preparation start times and pickup schedules
- [Driver Assignment] System selects driver experienced with multi-restaurant orders and provides detailed pickup route with timing requirements

Alternative Flows:

- [Restaurant Unavailability] If one restaurant becomes unavailable after order placement, system offers substitution options or partial order fulfillment with adjusted pricing
- [Timing Coordination Failure] If restaurants cannot synchronize preparation times, system offers staggered delivery option or order cancellation with full refund
- [Driver Capacity Issues] If no drivers are available for complex multi-restaurant route, system may split order among multiple drivers or reschedule delivery

Reflection Document

How did we decide what NOT to do?

In designing our MVP, we prioritized use cases that directly support the **core customer journey** (menu → order → delivery) and highlight our **differentiating features** (Nutrition & Allergen Transparency Engine and AI Food Assistant). The included features ensure that customers can browse menus, place orders, and receive deliveries while experiencing the unique value of nutrition transparency, something contemporary apps do not emphasize.

We chose **not to implement advanced financial flows (payments, refunds, commissions, payout cycles)**, **dynamic pricing models (AI-driven or surge)**, **deep regulatory/compliance features (tax reporting, illness logs, audits)**, and **secondary innovations (reusable containers, age verification)** at this stage. These were excluded because:

- They are **not critical to proving the MVP's viability** (users can still order and receive food).
- They **require specialized integrations or regulatory approval** that do not advance our core validation goals.
- They can be **layered on later** without disrupting or redesigning core workflows.

This approach allows us to deliver a lean but functional system, while leaving room for expansion by future teams.

What negative impacts or disappointments could this MVP have for stakeholders?

- **Customers:**
    - May be disappointed that payments, refunds, or loyalty rewards are not available.
    - Some may expect advanced delivery features (real-time map tracking, ETA prediction) which are absent in the MVP.
- **Restaurants:**
    - Lack of commission/payout handling could frustrate partners who want immediate financial reporting.
    - Sustainability-focused restaurants may be disappointed that reusable container flows are deferred.
- **Delivery Drivers:**
    - No advanced route optimization or surge pricing means less efficiency and fewer earning optimization opportunities.
- **Regulators/Health Authorities:**
    - Might expect illness reporting, compliance audits, or tax workflows; since these are excluded, the platform cannot yet demonstrate regulatory robustness.

While these omissions may disappoint certain stakeholders, we consider them acceptable trade-offs because the MVP still demonstrates clear customer value while testing our differentiator.

What changes did we make (and why) to appease at least some stakeholders?

- **Health-conscious Customers & Regulators:**
    - We included the **Nutrition & Allergen Transparency Engine** as a core feature, ensuring food is labeled with allergens, nutrition breakdowns, and healthier preference filters. This directly addresses a major gap in current apps.
- **Restaurant Stakeholders:**

  - While payouts and commissions are excluded, we enabled **menu management with discounts** and tied menu items to nutrition transparency. This gives restaurants immediate tools to attract health-oriented customers and validate differentiation.
- **Future Team Handoff:**
  - We intentionally left “hooks” in the schema and APIs for **payments, refunds, and sustainability features**, so another team can extend the platform without rework.

Conclusion

Our MVP design deliberately sacrifices **breadth of features** for **depth of innovation**. By focusing on ordering, delivery, and nutrition transparency, we maximize learning about customer adoption of our unique features. Although some stakeholders may be disappointed by missing financial, regulatory, or advanced logistics features, these were excluded strategically to keep the MVP lean while still addressing the most pressing and distinctive value propositions.

Prompt History

1)

For the purpose of designing the initial look of our application landing page, we used the help of chatgpt to create a design image with both zero shot prompting and careful prompting
Below is the prompt given:

Zero-shot prompting: *Create a website page design image with the below details 1st - Landing Page - The name of the cafe - 'WolfCafe' and the login form with the login button below the form - where there is an option for entering the username 'USERNAME' and password 'PASSWORD' in the form and option to select the role with which you want to login - menu manager , kitchen/inventory manager, customers - 'Choose your role'*
Below is the design generated:

Careful prompting (few shot prompting): *Create a website page design image where The theme of the design should be red, white and black with the below details*
1st - Landing Page - The name of the cafe - 'WolfCafe' (with a howling wolf in

a circular border image as the logo image)
and the login form with the login button below the form - where there is an option for
- entering the username 'USERNAME' , with small reference image
- password 'PASSWORD', with small reference image
- option to select the role with which you want to login 'CHOOSE YOUR ROLE: '
- menu manager , kitchen/inventory manager, customers, delivery manager...with a small reference image for each of the options option menu manager, kitchen/inventory manager, customer, delivery manager
Below is the design generated:

2)

Prompt Given to the gpt-5 rag model for generating the use case:

"""Generate 5 new use cases for a food delivery system that are not duplicates of already generated ones.
Use the following example to understand template and follow it strictly:
Use Case: Car Driving at Intersection
Description: -- Car approaches at intersection.
Preconditions: -- the traffic light has been initialized.
Main Flow:
1. The driver approaches the intersection.
2. The driver checks the status [Check Status].
3. The driver clears the intersection [Go].
Subflows:
• [Check Status] The driver checks the light [Check Light] and the queue [Check Queue]. If the light is
green and the queue is empty the driver clears the intersection [Go]. Otherwise, the driver joins a
queue [Join Queue].
• [Check Light] Check if the light is red, yellow, or green.
• [Check Queue] Check if the queue is empty or not.
• [Go] The driver clears the intersection and the use case ends.
• [Join Queue] The driver joins the end of the queue and checks status every 15 seconds
Alternative Flows:
• [Light Out] The light is not turned on. Wait for a clear intersection and gun

it.

- *[Accident] An accident is blocking the intersection. Rubber neck and slowly drive around it.*
- *[Ice Storm] There is an ice storm preventing safe driving. Abandon your car and walk.*

Context documents:
{ The documents that was shared with us for the reference }

Make sure each use case is realistic and relevant to food delivery systems.
"""

To make sure that we are getting some new Use Cases, we also provided this prompt

*"""You are an assistant that generates **new, non-duplicate use cases** for a food delivery app*
Avoid repeating any use case from this list:
Use Case: Schedule a Food Order
Use Case: Contactless Delivery
Use Case: Handling Out-of-stock Item.
Use Case: Handling Missing Item.
Use Case: Edits/Cancel Order
Use Case: Payment System
Use Case: Reward System
Use Case: Allergen Safe Order
Use Case: Transparency Report on Nutrition ranking
Use Case: Assign a Delivery Driver
Use Case: Refund System
Use Case: Restaurant Staff Updates Menu Availability
Use Case: Customer Reports Order Issue and Requests Resolution
Use Case: Staff Manages Real-Time Inventory Updates
Use Case: System Generates Dynamic Delivery Route Optimization
Use Case: Staff Manages Inventory Levels
Use Case: Admin Analyzes Delivery Performance Metrics
Use Case: Running payout cycles
Use Case: Enabling Healthier Preferences.
Use Case: Remit Transaction Taxes
Use Case: Refilling consumer-owned containers at pickup

Use Case: Bare-Hand Contact Program
Use Case: Sustainability-Driven Delivery Model"""

3)

To select our Use Cases from all the Use Cases that we have described as a part of a1,b1 and c1, we have used the following prompt in chatgpt

*These are the use cases that we have written as a part of the food delivery application that we are developing Use Case: Schedule a Food Order Use Case: Contactless Delivery Use Case: Handling Out-of-stock Item. Use Case: Handling Missing Item. Use Case: Edits/Cancel Order Use Case: Payment System Use Case: Reward System Use Case: Allergen Safe Order Use Case: Transparency Report on Nutrition ranking Use Case: Assign a Delivery Driver Use Case: Refund System Use Case: Restaurant Staff Updates Menu Availability Use Case: Customer Reports Order Issue and Requests Resolution Use Case: Staff Manages Real-Time Inventory Updates Use Case: System Generates Dynamic Delivery Route Optimization Use Case: Staff Manages Inventory Levels Use Case: Admin Analyzes Delivery Performance Metrics Use Case: Running payout cycles Use Case: Enabling Healthier Preferences. Use Case: Remit Transaction Taxes Use Case: Refilling consumer-owned containers at pickup Use Case: Bare-Hand Contact Program Use Case: Sustainability-Driven Delivery Model Use Case: Collaborative Group Ordering Use Case: Curbside Pickup Handoff Use Case: Tamper-Evident Packaging Verification Use Case: Secure Building Access Code Exchange Stakeholder : Menu Manager * Use Case : Add a new Item on the menu * Use Case : Update a Item on the menu * Use Case : Deleting a Item from the menu * Use Case : Apply Discounts on item/s in the menu Stakeholder: Inventory/Kitchen Manager * Use Case : Order Received * Use Case : Order Ready for Pickup * Use Case : Update an Item is Out of Stock Stakeholder: Customers * Use Case : Add an item to the cart * Use Case : Remove an item to the cart * Use Case : Place the order Stakeholder: Delivery Manager/Driver * Use Case: Pickup food * Use Case: Deliver food Stakeholder: Nutritionist * Use Case: Suggest a new healthy menu item. * Use Case: Ingredient Removal Select some use cases (more than 10 at least of all the above) which directly support the core customer journey (menu → order → delivery) and highlight our differentiating features (Nutrition & Allergen Transparency Engine and AI Food Assistant). The included features should ensure that customers can browse menus, place orders, and receive deliveries while experiencing the unique value of nutrition transparency, something contemporary apps do not emphasize. Please do not choose to implement advanced financial flows (payments, refunds, commissions, payout* *cycles), dynamic pricing models (AI-driven or surge), deep regulatory/compliance features (tax reporting, illness logs, audits), and secondary innovations (reusable containers)*

Some of the additional prompt that we used to generate refined solutions from RAG ChatGPT Based model are below: —

Context: We told ChatGPT to cluster the whole data into different categories such as Nutrition & Health, Business & Taxation, Technology + software, ,Business models + market studies.

Based on this we got the below prompts : —

Nutrition & Health: —

Generate NEW and DISTINCT use cases for a restaurant and food service ecosystem related to nutrition programs, dietary guidelines, or health compliance. Cover interactions between customers, nutritionists, and digital ordering platforms.

Business & Taxation: —

Generate NEW and DISTINCT use cases for a restaurant and food service ecosystem related to financial operations, taxation, and compliance. Include flows for restaurant owners, delivery partners, and third-party platforms.

Technology & Software : —

Generate NEW and DISTINCT use cases for a restaurant and food service ecosystem focusing on technology and software engineering. Cover mobile app development, microservices, interface design, and enterprise system integration.

User Experience & Psychology : —

Generate NEW and DISTINCT use cases for a restaurant and food service ecosystem related to customer behaviors, adoption drivers, trust, and satisfaction. Include different stakeholder perspectives (customer, delivery staff, restaurant manager).

Market Studies & Case Studies :—

Generate NEW and DISTINCT use cases for a restaurant and food service ecosystem based on case studies and market dynamics. Include flows around expansion, pandemic-era adaptations, restaurant–platform relationships, and regional adoption.

From the above prompts, we were able to generate many different use cases, and on applying further more restrictions like explicitly telling model to not generate use cases from particular set of topics, variety of use cases were explored.

Commercial systems using Claude:

1) Zero-shot prompts

https://claude.ai/share/ad5a02bf-6102-41c1-9985-35989f3fc1b8

https://claude.ai/share/55597280-b490-4422-aa9d-9541cdb965d0

2) Few-shot prompts (careful prompts)

https://claude.ai/share/17b14cb4-f99f-4c1f-8937-883044713d68

https://claude.ai/share/98b41550-7b6f-43af-89ed-1c9cf1d0e09d

https://claude.ai/share/782b3b45-ab04-4040-a1cd-d1b488400550

RAG implementation using Claude API:"do it yourself" local LLM

1) What is a use case?

Input: system="You are a helpful assistant. Use the provided context to answer the user's question. If you cannot find the answer in the context, state that you cannot answer."

Output:

Definition:
A sequence of transactions performed by a system that yields a measurable result of values for a particular actor/user.

Key Characteristics:

- Represents a major piece of functionality that is complete from beginning to end
- Must deliver something of value to an actor
- Describes external system behavior from the user's perspective

- Represents a complete course of events in the system

Purpose:

- Illustrates the system's intended functions from the user's perspective
- Focuses on how a system will achieve a goal for a user
- Shows how a system will be used and why the system is needed

Components:

- **Basic flow –** steps that capture the interaction between user and system
- **Alternative flows –** other ways of using the system or problem paths
- Includes both successful/happy paths and problem/alternative paths

In Use Case Diagrams:

- Represented as ovals
- Show relationships between use cases and actors (stick figures)
- The collection of all use cases represents everything that can be done to/with the system

Use cases are written by developers, fully documented and archived, and are read by developers, stakeholders, and customers to understand system functionality and requirements.

2) **Can you generate use cases about food delivery system?**

Input: system="You are a helpful assistant. Use the provided context to answer the user's question. If you cannot find the answer in the context, state that you cannot answer."

Output:

I cannot find information about food delivery system use cases in the provided context. The context primarily discusses food safety regulations, container refilling policies, and some brief mentions of consumer behavior research related to food delivery apps like Go-food and Grab-food, but it does not contain specific use cases for food delivery systems.

To generate meaningful use cases for a food delivery system, I would need context that includes information about system requirements, user interactions, functional specifications, or existing use case examples for food delivery platforms.

3) Generate 3 use cases about food delivery system.

Implementation: system=”You are a helpful assistant specializing in system analysis and use case generation. The user has provided some context from their documents, but it may not directly contain what they're asking for. First, briefly acknowledge what relevant information (if any) you found in the context. Then, use your expertise to provide a helpful response to their question, drawing inspiration from the context where possible but not limited to it. Be clear about what comes from the provided context versus your general knowledge.”

Sample Output:

Use Case 1: Customer Food Ordering Process

Actor: Customer
Description: A customer uses the food delivery app to browse restaurants, select food items, and place an order.
Steps:

1. Customer opens the food delivery app (Go-food or Grab-food as mentioned in context)
2. Customer browses available restaurants and food options
3. Customer selects desired food items and adds them to cart
4. Customer reviews order and proceeds to checkout
5. Customer provides delivery address and payment information
6. System confirms order and provides estimated delivery time
7. Customer receives order confirmation and tracking information

4) Generate 3 use cases about food delivery system with better use case format

Input:

use_case_prompt = f"""You are a requirements engineer specializing in use case documentation. Generate {num_cases} use cases for a {domain} following the exact template format shown below.
REQUIRED FORMAT EXAMPLE for each use case:

Use Case: Staff Manages Real-Time Inventory Updates
Description:

Restaurant staff updates item availability and inventory levels in real-time to prevent overselling and maintain accurate menu displays for customers.
Preconditions:
- Staff member is authenticated with inventory management permissions
- Restaurant is actively receiving orders through the system
- Menu items have established inventory tracking parameters
- System connectivity is stable
Main Flow:
1. Staff accesses inventory management dashboard during service hours
2. Staff reviews current stock levels and pending order queue [Check Current Status]
3. Staff identifies items running low or out of stock [Inventory Assessment]
4. Staff updates item availability and quantities in the system [Update Inventory]
5. System automatically adjusts menu displays and notifies affected customers [Propagate Changes]
Subflows:
- [Check Current Status] Staff views real-time inventory levels, preparation times, and order volume metrics
- [Inventory Assessment] Staff evaluates remaining portions, ingredient availability, and kitchen capacity constraints
- [Update Inventory] Staff marks items as unavailable, reduces quantities, or adjusts preparation times
- [Propagate Changes] System removes unavailable items from customer view and updates estimated delivery times

Alternative Flows:
- [Emergency Stock Depletion] If popular item suddenly runs out during peak hours, system offers automatic substitution suggestions to pending orders
- [Supplier Delivery Received] If new inventory arrives mid-service, staff can quickly restore item availability and notify waitlisted customers
- [Kitchen Equipment Failure] If equipment breaks down affecting specific menu categories, staff can bulk-disable related items and activate maintenance mode notifications"""

Sample Output:

Use Case: Customer Places Multi-Restaurant Order with Delivery Coordination
Description:
A customer orders items from multiple restaurants within the same delivery

zone, and the system coordinates pickup timing and delivery routing to ensure all items arrive together at optimal temperature.

Preconditions:

- Customer has valid account with verified delivery address
- Multiple participating restaurants are available in customer's delivery zone
- Customer has valid payment method on file
- Delivery drivers are available for coordinated pickup service
- System can calculate optimal routing and timing algorithms

Main Flow:

1. Customer browses available restaurants and adds items from multiple establishments to cart
2. System calculates preparation times and coordinates delivery windows [Calculate Coordination Timeline]
3. Customer reviews consolidated order summary with estimated delivery time and total cost
4. Customer confirms order and payment is processed for all restaurants simultaneously [Process Multi-Vendor Payment]
5. System dispatches preparation requests to each restaurant with staggered timing [Coordinate Restaurant Preparation]
6. System assigns delivery driver and optimizes pickup route based on preparation completion times [Optimize Delivery Route]
7. Driver collects orders from each restaurant following the coordinated schedule
8. System tracks delivery progress and notifies customer of consolidated delivery completion

Subflows:

- [Calculate Coordination Timeline] System analyzes each restaurant's preparation time, distance between locations, and optimal delivery window to synchronize completion
- [Process Multi-Vendor Payment] System splits payment appropriately between restaurants, applies delivery fees, and processes single

transaction for customer

- [Coordinate Restaurant Preparation] System sends timed preparation requests ensuring all orders complete within 5-minute window of each other
- [Optimize Delivery Route] System calculates most efficient pickup sequence and assigns driver with appropriate vehicle capacity

Alternative Flows:

- [Restaurant Preparation Delay] If one restaurant experiences delays, system automatically adjusts other restaurants' start times and notifies customer of revised delivery estimate
- [Driver Unavailable for Multi-Pickup] If no drivers can handle multiple pickups, system offers option to split into separate deliveries or cancel specific restaurant items
- [Payment Failure for One Vendor] If payment fails for one restaurant, system allows customer to remove those items or update payment method while preserving other restaurant orders

5) Generate 5 new use cases for food delivery system that are not mentioned in Project_1b1.pdf

Input:

existing_use_cases = [
"Add a new Item on the menu",
"Update a Item on the menu",
"Delete a Item from the menu",
"Order Received (kitchen preparation)",
"Order Ready for Pickup",
"Suggest a new healthy menu item",
"Ingredient Removal",
"Schedule a Food Order",
"Staff Manages Real-Time Inventory Updates",
"System Generates Dynamic Delivery Route Optimization",
"Reward System",
"Customer Reports Order Issue and Requests Resolution"
]

new_use_cases_prompt = f"""You are a senior requirements engineer working on a food delivery system.

EXISTING USE CASES TO AVOID (already documented in Project_1b1.pdf):
{chr(10).join(f"- {uc}" for uc in existing_use_cases)}

Generate {num_cases} COMPLETELY NEW AND DIFFERENT use cases that focus on these unexplored areas:

1. Driver account management and onboarding
2. Restaurant analytics and performance monitoring
3. Customer loyalty and membership programs
4. System security and fraud prevention
5. Multi-restaurant order coordination

Each use case must be COMPLETELY DIFFERENT from existing ones and cover functionality NOT mentioned in Project_1b1.pdf.

Follow this EXACT format for each use case:

Use Case: Staff Manages Real-Time Inventory Updates
- Preconditions:
• Staff member is authenticated with inventory management permissions
• Restaurant is actively receiving orders through the system
• Menu items have established inventory tracking parameters
• System connectivity is stable

- Main Flow:
1. Staff accesses inventory management dashboard during service hours
2. Staff reviews current stock levels and pending order queue [Check Current Status]
3. Staff identifies items running low or out of stock [Inventory Assessment]
4. Staff updates item availability and quantities in the system [Update Inventory]
5. System automatically adjusts menu displays and notifies affected customers [Propagate Changes]

- Subflows:
• [Check Current Status] Staff views real-time inventory levels, preparation times, and order volume metrics
• [Inventory Assessment] Staff evaluates remaining portions, ingredient

availability, and kitchen capacity constraints
• [Update Inventory] Staff marks items as unavailable, reduces quantities, or adjusts preparation times
• [Propagate Changes] System removes unavailable items from customer view and updates estimated delivery times

- Alternative Flows:
• [Emergency Stock Depletion] If popular item suddenly runs out during peak hours, system offers automatic substitution suggestions to pending orders
• [Supplier Delivery Received] If new inventory arrives mid-service, staff can quickly restore item availability and notify waitlisted customers
• [Kitchen Equipment Failure] If equipment breaks down affecting specific menu categories, staff can bulk-disable related items and activate maintenance mode notifications

Requirements:

- Focus on functionality NOT covered in existing use cases
- Each use case should represent a complete user journey
- Include realistic error handling and edge cases
- Make each use case practical and implementable”””

system="You are a professional requirements engineer with expertise in food delivery systems. Generate comprehensive, realistic use cases that would be suitable for academic coursework and real-world implementation. Ensure each use case is distinct and covers different system functionality.”

Sample Output:

Use Case 1: Driver Completes Account Verification and Onboarding

Preconditions:
• Prospective driver has submitted initial application through mobile app
• Driver possesses valid driver's license, vehicle registration, and insurance documentation
• Background check and vehicle inspection requirements are defined in system
• Onboarding workflow is configured with required verification steps

Main Flow:

1. Driver receives onboarding notification and accesses verification portal

2. Driver uploads required documentation (license, insurance, vehicle photos) [Document Submission]
3. System validates document authenticity and completeness [Document Verification]
4. Driver completes vehicle inspection at approved location [Physical Verification]
5. System processes background check and insurance verification [Background Processing]
6. Driver completes platform training modules and safety certification [Training Completion]
7. System activates driver account and enables order acceptance capability

Subflows:

- [Document Submission] Driver photographs and uploads clear images of all required documents with system validating image quality and readability
- [Document Verification] System performs OCR scanning, cross-references with DMV databases, and flags any discrepancies for manual review
- [Physical Verification] Approved inspector verifies vehicle condition, safety equipment, and compliance with delivery standards
- [Background Processing] System interfaces with third-party services to verify criminal history, driving record, and insurance coverage validity
- [Training Completion] Driver watches safety videos, completes quiz modules, and acknowledges platform policies and procedures

Alternative Flows:

- [Document Rejection] If uploaded documents are unclear or expired, system provides specific feedback and allows re-submission with guidance
- [Failed Background Check] If background check reveals disqualifying issues, system sends rejection notification with appeal process information
- [Insurance Lapse] If insurance verification fails, system suspends onboarding and provides instructions for coverage correction