Requirement Analysis

Hardik Majethia, Pranshav Gajjar,
Aniruddh Sanjeev Bhagwat, Ishan Patel

September 8, 2025

UC-1: Order Pickup and Delivery

Preconditions: The Rider is assigned to the order [**Assign Rider**]. The food is marked as "Ready for Pickup" by the restaurant.

Main Flow:

1. The Rider arrives at the restaurant [**Arrive Restaurant**].
2. The Rider picks up the order [**Pickup Food**].
3. The Rider travels to the customer's address [**Navigate Route**].
4. The Rider delivers the food to the customer [**Deliver Food**].

Subflows:

- [**Arrive Restaurant**]: The Rider navigates to the restaurant location and indicates their arrival through the app.
- [**Pickup Food**]: The restaurant staff verifies the order ID with the Rider before handing over the packaged food. The Rider confirms pickup in the app.
- [**Navigate Route**]: The app provides an optimized GPS route to the customer's delivery address.
- [**Deliver Food**]: The Rider hands the order to the customer and marks the order as "Complete" in the app.

Alternative/Error Flows:

- If the Rider cannot find the restaurant, they can contact the restaurant or support through the app. If unresolved, the order may be reassigned.
- If the customer is unavailable at the delivery address, the Rider follows the standard procedure for failed delivery (e.g., contact customer, wait for a specified time, contact support).

UC-2: Manage Allergen Information and Filtering

Preconditions: A Restaurant Partner is onboarded on the platform. A new menu item is added by the restaurant OR a customer is viewing a menu.

Main Flow:

1. The system requires the restaurant to declare potential allergens for the new menu item [**Declare Allergens**].
2. The system displays the allergen information prominently on the menu for the customer [**Display Allergens**].
3. The customer applies an allergen filter (e.g., "No Nuts") to the menu [**Apply Filter**].
4. The system displays only items that comply with the filter [**Display Filtered Menu**].

Subflows:

- [**Declare Allergens**]: The restaurant must select from a predefined list of major allergens (e.g., nuts, dairy, shellfish) for each menu item. The app does not allow the item to be saved without this step.
- [**Display Allergens**]: The customer's app UI shows allergen icons and text warnings on the menu and in the cart before checkout.
- [**Apply Filter**]: The customer selects one or more allergens to exclude from their menu view.
- [**Display Filtered Menu**]: The system cross-references the restaurant's allergen declarations and removes any non-compliant items.

Alternative/Error Flows:

- If a restaurant fails to declare allergens, the system flags the menu item as "Non-Compliant" and hides it from customer view until the information is provided.
- If a customer has a severe allergy and indicates it in a special instruction, the system can display an additional warning recommending they contact the restaurant directly before ordering.

UC-3: Manage Restaurant Health and Safety Compliance

Preconditions: A new restaurant applies to join the platform OR a restaurant is live on the platform for more than 6 months.

Main Flow:

1. The system requires the restaurant to upload a valid food safety certification document [**Upload Certificate**].
2. A compliance officer reviews the document for validity [**Verify Certificate**].
3. Upon verification, the restaurant onboarding proceeds [**Approve Onboarding**].
4. The system automatically schedules a periodic virtual audit [**Schedule Audit**].
5. The system requests the restaurant to submit recent health inspection reports and photos of food preparation areas [**Request Evidence**].
6. A compliance officer reviews the evidence [**Review Evidence**].

Subflows:

- [**Upload Certificate**]: The app provides a secure portal for document upload.
- [**Verify Certificate**]: The officer checks the certificate's issuing authority, expiry date, and restaurant details.
- [**Request Evidence**]: The app sends a notification to the restaurant with a list of required evidence and a deadline.
- [**Review Evidence**]: The officer assesses the evidence against platform health standards.

Alternative/Error Flows:

- If the certificate is expired or invalid, the system notifies the restaurant and pauses the onboarding process until a valid certificate is provided.
- If the restaurant fails to provide evidence or fails the audit, the system can place a "Under Review" badge on their listing or suspend them until compliance is met.

UC-4: Facilitate Contactless Delivery

Preconditions: A global health protocol (e.g., pandemic) mandates reduced contact. An order is placed and assigned to a rider.

Main Flow:

1. The system prompts the customer to choose a delivery option: "Leave at Door" or "Hand to Me" [**Select Delivery Preference**].
2. The system communicates the customer's preference to the Rider [**Communicate Preference**].
3. The Rider executes the delivery as preferred [**Execute Delivery**].

Subflows:

- [**Select Delivery Preference**]: This option is presented during checkout.
- [**Communicate Preference**]: The chosen method is displayed prominently on the Rider's app order screen.

Alternative/Error Flows:

- If the customer selects "Leave at Door" but is in a secure apartment building, the Rider follows a procedure to contact the customer for instructions or leave the order in a designated safe area.

UC-5: Manage Nutritional Information Display

Preconditions: Local regulations mandate calorie labeling on menus. A restaurant partner operates in a regulated region.

Main Flow:

1. The system requires restaurants in regulated regions to input calorie counts for standard menu items [**Input Calorie Data**].
2. The system displays calorie information next to the item name and price on the customer's app [**Display Calorie Info**].

Subflows:

- [**Input Calorie Data**]: A mandatory field is added to the restaurant's menu management portal for items available in regulated regions.
- [**Display Calorie Info**]: The customer app UI is updated to show "xxx kcal" in a standardized format.

Alternative/Error Flows:

- If a restaurant does not provide calorie data for a regulated region, the system prevents that specific menu item from being sold in that region.

UC-6: Restrict Price Promotions on Unhealthy Foods

Preconditions: A public health policy restricts promotions on HFSS (High in Fat, Salt, and Sugar) foods. The platform has a promotion engine.

Main Flow:

1. The system tags menu items classified as HFSS based on restaurant-provided nutritional data [**Tag HFSS Items**].
2. When creating a promotion, the system blocks discounts from being applied to tagged HFSS items [**Enforce Promotion Rules**].

Subflows:

- [**Tag HFSS Items**]: Items are automatically tagged based on an algorithm using declared nutritional values against a defined nutrient profile model.
- [**Enforce Promotion Rules**]: The promotion management tool will not allow a percentage-off or BOGOF (Buy One Get One Free) promotion to include HFSS-tagged items.

Alternative/Error Flows:

- If a restaurant attempts to circumvent the rule by creating a new "healthy" menu item that is actually HFSS, the system flags it for manual review by the compliance team.
- If nutritional data is missing or appears fraudulent, the system flags the item for manual review and verification by a compliance officer before the HFSS tag is applied.

UC-7: Handle Food Tampering Complaint

Preconditions: An order has been delivered. A customer submits a complaint alleging package tampering.

Main Flow:

1. The customer submits a complaint with photo evidence through the app [**Submit Complaint**].
2. The system immediately flags the order and the involved Rider for review [**Flag for Review**].
3. A support agent investigates the complaint by reviewing order tracking and Rider history [**Investigate Incident**].

Subflows:

- [**Submit Complaint**]: The app provides a dedicated path for "Safety Concerns" with an option to upload images.
- [**Investigate Incident**]: The agent reviews the GPS trail of the Rider and any previous similar complaints.

Alternative/Error Flows:

- If the complaint is verified, the system blocks the Rider from receiving new orders pending further investigation and initiates a refund/replacement for the customer.

UC-8: Onboard Restaurant for WIC Program

Preconditions: The USDA WIC program allows online ordering (as proposed). A grocery store on the platform is WIC-authorized.

Main Flow:

1. The system enables a WIC-authorized grocery store to tag eligible items in their inventory [**Tag WIC Items**].
2. The system configures the store's checkout to accept WIC EBT as a payment method [**Enable WIC Payment**].

Subflows:

- [**Tag WIC Items**]: The store's admin portal has a function to mark items that are WIC-approved. The system validates these against an official product lookup table.
- [**Enable WIC Payment**]: The payment processing module is integrated with the WIC EBT system for online transactions.

Alternative/Error Flows:

- If a customer attempts to purchase a non-WIC item with their WIC benefits, the system flags the error at checkout and prevents the transaction from proceeding.

UC-9: Process WIC EBT Transaction

Preconditions: A customer has a WIC EBT card registered in the app. The customer's cart contains only WIC-eligible items.

Main Flow:

1. The customer selects "WIC EBT" as their payment method at checkout [**Select Payment**].
2. The system validates that every item in the cart is WIC-eligible [**Validate Cart**].
3. The system processes the payment through the WIC EBT gateway [**Process WIC Payment**].

Subflows:

- [**Validate Cart**]: The system cross-references each cart item with the store's tagged WIC-eligible items. If any item is ineligible, the transaction is halted.
- [**Process WIC Payment**]: The app connects to the state's WIC EBT processing system to authorize and complete the benefit redemption.

Alternative/Error Flows:

- If the WIC EBT system is offline, the transaction fails and the customer is notified to try again later or use an alternative payment method for non-WIC items.

UC-10: Calculate Tax and Generate Compliant Invoice

Preconditions: An order is placed by a customer.

Main Flow:

1. The system determines the correct tax rules based on the delivery location, item type, and customer type (e.g., B2B) [**Determine Tax Jurisdiction**].
2. The system calculates the tax amount [**Calculate Tax**].
3. The system generates a detailed invoice including tax amount, breakdown, and any required legal information [**Generate Invoice**].

Subflows:

- [**Determine Tax Jurisdiction**]: The platform's tax engine uses the delivery address and item categorization to determine the correct tax rates.
- [**Calculate Tax**]: The engine applies the rates to the taxable subtotal.
- [**Generate Invoice**]: The invoice format includes all legally required fields for the jurisdiction (e.g., business name and tax ID for B2B, GST/HST breakdown for Canada).

Alternative/Error Flows:

- For Canadian orders, the system applies GST/HST and PST rates based on the delivery province and item type.
- For B2B orders within the EU where reverse charge applies, the system applies a 0% VAT rate after validating the customer's VAT ID through the VIES system and notes "Reverse Charge" on the invoice.
- If a customer's business tax information is incomplete, the system blocks checkout for a B2B order.

UC-11: Manage Sales Tax Exemption for Non-Profit Organizations

Preconditions: A registered non-profit organization has an account on the platform. The organization provides a valid tax exemption certificate.

Main Flow:

1. The organization uploads their tax exemption certificate to their profile [**Upload Exemption Cert**].
2. The platform's compliance team verifies the certificate [**Verify Certificate**].
3. Once verified, the system applies a tax exemption to all orders placed by this organization [**Apply Exemption**].

Subflows:

- [**Upload Exemption Cert**]: A secure portal is provided for document upload.
- [**Verify Certificate**]: A compliance officer reviews the document for validity, expiry date, and jurisdiction.
- [**Apply Exemption**]: The tax calculation engine checks the customer's tax-exempt status before applying taxes at checkout.

Alternative/Error Flows:

- If the certificate expires, the system automatically reverts to charging sales tax and notifies the organization to upload a renewed certificate.

UC-12: Report Earnings to Tax Authorities (Rider)

Preconditions: The tax year has ended. A Rider has earned above the tax reporting threshold on the platform. The Rider's tax information (e.g., SSN) is on file and valid.

Main Flow:

1. The system compiles all earnings and incentive data for the Rider for the tax year [**Compile Earnings**].
2. The system generates a formal tax document (e.g., 1099-NEC in the US) [**Generate Tax Form**].
3. The system makes the tax document available for the Rider to download from their app and files it with the relevant tax authority [**Issue and File Form**].

Subflows:

- [**Compile Earnings**]: The platform's database aggregates all payments, bonuses, and reimbursements for the user.
- [**Issue and File Form**]: The Rider receives a push notification that their tax document is ready. The platform electronically files the information with the IRS (or equivalent).

Alternative/Error Flows:

- If a Rider's tax information is missing or invalid, the system withholds future earnings until the information is corrected to ensure compliance.

UC-13: Deduct Withholding Tax for International Riders

Preconditions: A Rider provides services in a country where they are not a tax resident. A tax treaty requires the platform to withhold tax. The Rider's tax residency information is on file.

Main Flow:

1. The system identifies a Rider's tax residency status based on their provided information and payment address [**Identify Status**].

2. The system applies the correct withholding tax percentage to their earnings as per the local regulations and relevant tax treaty [**Apply Withholding**].
3. The system remits the withheld tax to the local tax authority [**Remit Tax**].

Subflows:

- [**Identify Status**]: The system checks the Rider's declared country of tax residency.
- [**Apply Withholding**]: The payment processing system calculates the withholding amount on each payout.

Alternative/Error Flows:

- If a Rider does not provide a valid Tax Identification Number (TIN) from their country of residency, the system may apply a higher default withholding rate.

UC-14: Automate Sales Tax Remittance (US Marketplace Facilitator Law)

Preconditions: The platform is considered a Marketplace Facilitator under US state law. An order is placed by a customer in a state with such laws.

Main Flow:

1. The platform's system calculates and collects sales tax from the customer at checkout [**Collect Tax**].
2. The platform is responsible for remitting the collected sales tax directly to the state tax authority [**Remit Tax**].
3. The restaurant partner is not liable for calculating or remitting this sales tax [**Relieve Restaurant**].

Subflows:

- [**Collect Tax**]: The tax engine uses the delivery address to determine the correct combined state and local sales tax rate.
- [**Remit Tax**]: The platform's finance system aggregates taxes collected by jurisdiction and files periodic electronic returns.

Alternative/Error Flows:

- The platform's tax engine subscribes to a certified tax rate update API to ensure rates are always current. A manual process exists for emergency updates if the API is unavailable.

UC-15: Handle Customer-Restaurant Tip Allocation

Preconditions: A customer adds a tip to their order. The tip is distributed between the restaurant and the Rider.

Main Flow:

1. The system allocates the tip amount between the Rider and the restaurant based on a pre-defined, legally compliant policy (e.g., 100% to Rider, or a specific split) [**Allocate Tip**].
2. The system reports the tip allocation accurately on the transaction records for both the Rider and the restaurant for their own income reporting [**Report Tips**].

Subflows:

- [**Allocate Tip**]: The allocation logic is configured in the payment processing system and is transparently disclosed to all parties.
- [**Report Tips**]: The Rider's app and the restaurant's portal show a clear record of tips earned per order.

Alternative/Error Flows:

- If local labor laws change regarding tip pooling or ownership, the allocation logic must be updated to remain compliant.

UC-16: Validate Tax Documentation for Restaurant Payouts

Preconditions: A restaurant is due to receive their weekly earnings payout.
Main Flow:

1. The system checks that the restaurant's tax information (e.g., W-9 form in the US) is on file and valid before processing the payout [**Validate Docs**].
2. If valid, the system processes the payout [**Process Payout**].
3. The system generates a yearly earnings summary for the restaurant for their tax filing [**Generate Summary**].

Subflows:

- [**Validate Docs**]: The system checks for the presence and expiry date of necessary tax forms.
- [**Generate Summary**]: The restaurant's admin portal provides a downloadable report of all earnings and commission fees for the year.

Alternative/Error Flows:

- If tax documents are invalid or missing, the system places the payout on hold and sends an alert to the restaurant to update their information.

UC-17: Collect Tax Identification Information

Preconditions: A new user (Rider or Restaurant) is undergoing onboarding.
Main Flow:

1. The system prompts the user to declare their status (e.g., individual, business, non-profit) [**Declare Status**].
2. Based on the status, the system requests the appropriate tax identification number (e.g., SSN, EIN, VAT Number) [**Request TIN**].
3. The user inputs their tax information [**Input TIN**].
4. The system validates the format of the information [**Validate Format**].

Subflows:

- [**Request TIN**]: The required field is presented to the user with examples of the required format.
- [**Validate Format**]: The system checks that the number provided matches the expected format for the given country and type.

Alternative/Error Flows:

- If the format is invalid, the system prompts the user to correct it before proceeding with onboarding. Full validation against government databases (like VIES) occurs later during transaction processing.

UC-18: Manage Digital Service Tax (DST) Compliance

Preconditions: The platform operates in a country that has enacted a Digital Service Tax (DST). The platform's revenues in that country exceed the DST threshold.
Main Flow:

1. The system calculates the DST based on revenues generated from restaurant commissions and delivery fees within the specific country [**Calculate DST Liability**].
2. The system generates a periodic DST report for the finance team [**Generate Report**].
3. The finance team files and pays the DST to the relevant tax authority [**File and Pay**].

Subflows:

- [**Calculate DST Liability**]: The financial reporting system segregates revenue by country and applies the DST rate to the taxable revenue.
- [**Generate Report**]: The report details the calculation for the audit trail.

Alternative/Error Flows:

- If new DST legislation is passed, the system must be reconfigured by the finance and engineering teams to correctly identify and calculate the new tax before its effective date.

UC-19: Handle Product Recall Notification

Preconditions: A food supplier issues a recall for a specific ingredient or product. A restaurant on the platform has used the recalled item in menu items.

Main Flow:

1. The system identifies all menu items across all restaurants that contain the recalled product [**Identify Affected Items**].
2. The system automatically and immediately removes the affected menu items from sale [**Remove Items**].
3. The system notifies the restaurant owners of the recall and the action taken [**Notify Restaurant**].
4. The system notifies customers who have recently ordered the affected items [**Notify Customers**].

Subflows:

- [**Identify Affected Items**]: The system cross-references the recalled product with restaurant menu ingredient declarations.
- [**Notify Customers**]: The app sends a push notification and email to affected customers, detailing the recall reason and recommended actions.

Alternative/Error Flows:

- If a restaurant disputes the recall mapping, they can contact support to provide evidence and have their menu items reinstated.

UC-20: Enforce Age Verification for Restricted Items

Preconditions: An order contains age-restricted items (e.g., alcohol, cutlery). The order is being delivered.

Main Flow:

1. The system prompts the customer to acknowledge they are of legal age during checkout [**Acknowledge Age**].
2. The system highlights the order to the Rider as requiring age verification [**Flag for Rider**].
3. Upon delivery, the Rider requests valid ID from the recipient to verify age [**Verify ID**].
4. The Rider confirms successful verification in the app [**Confirm Verification**].

Subflows:

- [**Verify ID**]: The Rider checks the photo ID to ensure the recipient meets the legal age requirement.
- [**Confirm Verification**]: The Rider's app requires a mandatory step to confirm delivery compliance before marking the order complete.

Alternative/Error Flows:

- If the recipient cannot provide valid ID or is underage, the Rider follows the procedure for a failed delivery and returns the restricted items to the restaurant. The customer is charged a restocking fee.

UC-21: Audit Trail for Regulatory Compliance

Preconditions: A regulatory body requests information for an audit. An action related to health or tax compliance has occurred on the platform.

Main Flow:

1. The system retrieves all relevant data logs for the specified entity (Restaurant, Rider) and time period [**Retrieve Logs**].
2. The system compiles the data into a standardized report format [**Compile Report**].
3. The compliance team reviews and submits the report to the authorities [**Submit Report**].

Subflows:

- [**Retrieve Logs**]: The system accesses logs for health certificate updates, allergen declarations, tax calculations, rider earnings, and promo restrictions.
- [**Compile Report**]: The data is formatted to clearly show compliance with specific regulations.

Alternative/Error Flows:

- If the requested data is too vast, the system allows for targeted filtering based on the audit's specific focus (e.g., "all allergen-related menu changes for Restaurant X in Q2").

UC-22: Manage Regional Health Regulation Variations

Preconditions: The platform operates in multiple regions with different health codes. A restaurant operates in multiple regions.

Main Flow:

1. The system identifies the health regulations applicable to the restaurant's operating regions [**Identify Regulations**].
2. The system presents the restaurant with a checklist of region-specific compliance requirements [**Present Checklist**].
3. The restaurant confirms compliance for each region [**Confirm Compliance**].

Subflows:

- [**Identify Regulations**]: The system uses the restaurant's provided operating addresses to determine the relevant health jurisdictions.
- [**Present Checklist**]: The checklist includes requirements for food handler certifications, equipment standards, and labeling laws that vary by region.

Alternative/Error Flows:

- If a restaurant cannot confirm compliance for a specific region, the system prevents them from listing their menu in that region until confirmation is provided.

UC-23: Process Charitable Donation Tax Receipts

Preconditions: The platform offers a round-up donation feature at checkout for a registered charity. A customer donates using this feature.

Main Flow:

1. The system records the donation amount linked to the customer's account and order [**Record Donation**].
2. At the end of the tax year, the system compiles the total donations made by each eligible customer [**Compile Donations**].
3. The system generates and issues an official tax receipt to the customer for their total annual donations [**Issue Tax Receipt**].

Subflows:

- [**Record Donation**]: The donation is recorded as a separate line item on the order receipt.
- [**Issue Tax Receipt**]: The receipt is generated in accordance with charity tax law and made available for download in the customer's account portal.

Alternative/Error Flows:

- If the donation amount for a user is below the legal threshold for issuing a receipt, the system does not generate one but still records the transaction for the charity's internal tracking.

UC-24: Handle Data Breach Notification

Preconditions: A security incident results in the potential exposure of user health or financial data.

Main Flow:

1. The system identifies the scope of the breach and the users affected [**Identify Scope**].
2. The system triggers automated notifications to all affected users as required by law (e.g., GDPR, CCPA) [**Notify Users**].
3. The system guides users through steps to secure their accounts, such as password resets [**Guide Remediation**].

Subflows:

- [**Notify Users**]: Notifications are sent via all available channels (email, app alert) and contain details of the breach and recommended actions.
- [**Guide Remediation**]: The app provides direct links to password reset and credit monitoring services if applicable.

Alternative/Error Flows:

- If the breach involves payment data, the system automatically invalidates stored payment methods for affected users and requires re-entry upon next purchase.

UC-25: Synchronize Offline Compliance Updates

Preconditions: A device running the Rider or Restaurant app loses internet connectivity. Compliance-related actions are taken while offline (e.g., Rider confirms age verification, restaurant marks item as out of stock).

Main Flow:

1. The app stores the compliance actions locally in a queue [**Queue Actions**].
2. Once connectivity is restored, the app automatically synchronizes the queued data with the central platform [**Sync Data**].
3. The platform processes the synchronized actions and updates all relevant systems [**Process Updates**].

Subflows:

- [**Queue Actions**]: The app continues to function for critical compliance steps, storing data locally.
- [**Sync Data**]: A background process automatically detects connectivity and pushes all pending actions.

Alternative/Error Flows:

- If synchronization fails for a specific action, the app will retry and then alert the user to manually resolve the issue if retries are unsuccessful.

UC-26: Manage Supplier Liability Insurance Verification

Preconditions: A new restaurant supplier is onboarded.

Main Flow:

1. The system requires the supplier to upload a certificate of insurance (COI) [**Upload COI**].
2. The system parses the COI to extract key details (provider, coverage amounts, expiry date) [**Parse COI**].
3. A compliance officer verifies the details against platform requirements [**Verify COI**].

Subflows:

- [**Parse COI**]: An integrated OCR (Optical Character Recognition) tool reads the document to auto-fill data fields.
- [**Verify COI**]: The officer ensures coverage is adequate and current.

Alternative/Error Flows:

- If the COI is expired or insufficient, the system prevents the supplier from going live and notifies them of the deficiency.

UC-27: Route Optimization for Perishable Goods

Preconditions: An order contains items highly sensitive to temperature or time (e.g., ice cream, sushi).

Main Flow:

1. The system tags the order as "Perishable" [**Tag Order**].
2. The routing algorithm prioritizes this order and calculates the most direct route from restaurant to customer [**Calculate Priority Route**].
3. The system assigns the order to a Rider with appropriate equipment (e.g., insulated bag) if available [**Assign Rider**].

Subflows:

- [**Tag Order**]: The system automatically tags orders based on menu item categorization.
- [**Calculate Priority Route**]: The algorithm minimizes delivery time for this specific order, even if it is not the most efficient for the Rider's batch.

Alternative/Error Flows:

- If no appropriately equipped Rider is available, the system may prompt the restaurant to use extra packaging or can warn the customer of potential delivery delays for quality assurance.

UC-28: Dynamic Disclaimer Management

Preconditions: A menu item requires a specific disclaimer (e.g., "consuming raw or undercooked meats may increase your risk of foodborne illness").

Main Flow:

1. The system maintains a list of disclaimers linked to ingredient tags (e.g., "raw", "undercooked", "alcohol") [**Maintain Disclaimer List**].
2. When a menu item is created or updated, the system automatically adds required disclaimers based on its ingredients [**Apply Disclaimer**].
3. The system displays the disclaimer clearly on the item's description in the customer app [**Display Disclaimer**].

Subflows:

- [**Apply Disclaimer**]: The system cross-references the item's ingredients and preparation method with the disclaimer rules.
- [**Display Disclaimer**]: The disclaimer text is displayed in a standardized format, such as italicized text below the description.

Alternative/Error Flows:

- If a new regulation requires a new disclaimer, the platform admin can add it to the list, and it will be automatically applied to all relevant existing menu items.

UC-29: Calculate Carbon Tax Surcharge

Preconditions: The platform operates in a region with a carbon tax on delivery services.
Main Flow:

1. The system calculates the estimated carbon emissions for the delivery route [**Calculate Emissions**].
2. The system calculates the corresponding carbon tax based on the emissions and the current tax rate [**Calculate Surcharge**].
3. The system adds the surcharge as a separate, labeled line item at checkout [**Apply Surcharge**].

Subflows:

- [**Calculate Emissions**]: The calculation uses the vehicle type and distance of the route.
- [**Apply Surcharge**]: The surcharge is clearly explained to the customer as a "Carbon Offset Fee" or similar.

Alternative/Error Flows:

- If the customer chooses a "green" delivery option (e.g., bicycle courier), the system waives the carbon tax surcharge.

UC-30: Process Subsidized Meal Programs

Preconditions: The platform integrates with a government subsidized meal program (e.g., for seniors or low-income families). A user is verified as eligible for the program.
Main Flow:

1. The user applies for program benefits through the app by providing eligibility documentation [**Apply for Benefits**].
2. The system verifies the user's eligibility with the government program's API [**Verify Eligibility**].
3. Once verified, the user receives a subsidy balance that can be applied to eligible food items [**Receive Balance**].
4. At checkout, the subsidy is applied automatically to the order total [**Apply Subsidy**].

Subflows:

- [**Verify Eligibility**]: The app connects to the official government service to confirm the user's status.
- [**Apply Subsidy**]: The payment processing system applies the subsidy balance before charging the user's primary payment method for any remaining balance.

Alternative/Error Flows:

- If the user's eligibility expires, the system disables the subsidy and notifies the user to re-apply.

Reflection Document: Differences between LLM Reports

When analyzing the output of LLMs for requirements generation, it became clear that simple prompting is not sufficient. With simple prompting, the responses lacked structure and did not include subflows or alternative/error flows. The descriptions of the use cases were also written in broad terms, which introduced ambiguity and made the requirements less actionable. Careful prompting was necessary to achieve higher quality and more useful results.

With careful prompting, Gemini produced all the required use cases and organized them into categories. This structured presentation made the output easier to read and navigate. However, many of the individual use cases lacked subflows and error flows, which are an important part of requirement engineering. Without them, the use cases feel incomplete, even if they are presented in a clean, organized way.

DeepSeek, by contrast, did not initially provide a structured response. Its use cases were not categorized, which made them somewhat harder to follow. However, DeepSeek consistently included subflows and error flows for every use case. This level of completeness made its output more useful from an engineering perspective. While Gemini's categories provided clarity, DeepSeek's detailed coverage of alternative paths ultimately made its output stronger.

Overall, we preferred DeepSeek's response because it provided more thorough requirements, even though it lacked the structured categories that Gemini introduced. This comparison highlights the tradeoff between clarity and completeness, and shows that careful prompting combined with model selection plays a key role in producing high-quality requirements.