

# Proj1c1

## Use Cases

### **UC1 Customer - Browse Restaurants by Location**

A customer searches for nearby restaurants using GPS or an entered address.

#### Preconditions

- Customer App open.
- Device with internet access.

#### Main Flow

1. The customer taps "Browse." [S1]
2. The system retrieves location information. [S2] [A1] [A2]
3. The system shows a restaurant list with restaurant info [S3]
4. The customer selects a restaurant. [S4]
5. The system loads menu and availability. [S5]

#### Subflows

[S1] Initiate location-based search from the home screen.

[S2] Use the customer-provided address if available, otherwise request GPS access if not already granted.

[S3] Display a restaurant list including ETAs, price ranges, and ratings.

[S4] Select a restaurant for details.

[S5] Load menu, availability, and basic details.

#### Alternative Flows

[A1] **Out of Zone:** If the customer is outside of the delivery area, show pickup zones.

[A2] **Unable to Locate:** Prompt customer to enter an address and return to Step 2.

### **UC2 Customer - Build Cart & Customize Items**

A customer browses a menu, customizes items, and adds them to their cart.

#### Preconditions

- **UC1**, The selected restaurant is open for business.
- Menu is loaded.

#### Main Flow

1. The customer selects an item. [S1]
2. The customer customizes and adds the item to the cart. [S2] [A1] [A2]
3. The system recalculates totals. [S3]
4. The customer repeats for more items. [S4]

#### Subflows

[S1] Select menu item from category view.

[S3] Customer customizes and clicks "Add to Cart."

[S4] Totals update automatically.

**[S5]** The customer may continue browsing and add multiple items or proceed to checkout.

#### **Alternative Flows**

**[A1] Add-on Sold Out:** Prompt customer to substitute or skip.

**[A2] Item Removed:** Notify as out of stock, don't add item to cart.

## **UC3 Customer - Checkout & Payment**

A customer reviews order details, selects payment, and completes checkout.

#### **Preconditions**

- **UC2** - Items have been added to cart

#### **Main Flow**

1. The customer reviews order details including items, address, and tip. **[S1] [A2] [A3]**
2. The customer selects a payment method. **[S2]**
3. The system calculates taxes/fees and pre-authorizes payment. **[S3] [A1]**
4. The customer confirms the order. **[S4]**
5. The system places order/s and sends confirmation. **[S5]**

#### **Subflows**

**[S1]** The customer is shown their order summary - items, total, instructions, address, and privacy/data-use notices.

**[S2]** Choose card, wallet, or split payment.

**[S3]** Calculate taxes/fees, run pre-authorization.

**[S4]** Customer clicks "Confirm Order."

**[S5]** System sends confirmation with order # and ETA.

#### **Alternative Flows**

**[A1] Payment Declined:** Retry or select another method.

**[A2] Backtrack From Cart:** Return customer to cart menu.

**[A3] Scheduled Order:** The customer can schedule an order for future delivery

## **UC4: Restaurant - Onboarding**

A restaurant signs up for the platform and gains access to the dashboard.

#### **Preconditions**

- Device with internet access.
- Restaurant App is open

#### **Main Flow**

1. The restaurant manager selects "Sign Up." **[S1]**
2. The restaurant manager provides basic account credentials **[S2]**
3. The restaurant manager fills out their business profile, compliance docs, and payout details. **[S3] [A1]**
4. The restaurant submits the application. **[S4]**

5. The system verifies and activates the account. **[S5] [A2]**

#### **Subflows**

- [S1]** Restaurant manager clicks “Sign Up.”
- [S2]** Basic account information such as email and password are entered
- [S3]** Enter name, address, ownership, licenses, tax/PIN, bank (ACH).
- [S4]** Store documents; show status “Under Review.”
- [S5]** On approval, send confirmation; enable dashboard access.

#### **Alternative Flows**

- [A1] Missing/Invalid Fields:** highlight errors, block submission.
- [A2] Verification Fail:** request re-submission or route to support.

## **UC5: Restaurant - Menu Management**

A restaurant manager creates and maintains menu items customers can order.

#### **Preconditions**

- **UC4** - Restaurant account approved and logged in.
- Device with internet access.

#### **Main Flow**

1. The restaurant manager clicks the “Manage Menu” button **[S1]**
2. The system displays categories and current items. **[S2]**
3. The restaurant manager adds or edits an item (name, price, description). **[S3]**  
**[A1]**
4. The restaurant manager sets availability and options/modifiers. **[S4] [A2]**
5. The system publishes updates to the customer storefront. **[S5]**

#### **Subflows**

- [S1]** Navigate: Dashboard → Manage Menu.
- [S2]** View categories; reorder via drag-and-drop.
- [S3]** CRUD item: name, description, photos, tax class, price, calories.
- [S4]** Configure availability windows, sold-out toggle, options (sizes/add-ons).
- [S5]** Validate and sync; confirm success banner.

#### **Alternative Flows**

- [A1] Validation error:** block save, highlight fields needing correction.
- [A2] Conflicting availability windows:** show conflict and propose merge.

## **UC6: Restaurant - Setting Store Availability**

A restaurant manages its store hours and sets temporary pauses

#### **Preconditions**

- **UC4** - Restaurant account approved and logged in
- Device with internet access.

#### **Main Flow**

1. The restaurant manager clicks the “Set Availability” button. **[S1]**

2. The system displays the weekly schedule and current status. [S2]
3. The restaurant manager edits availability [S3] [S4]
4. The system updates storefront visibility and ETAs. [S5]

#### **Subflows**

[S1] Navigate: Settings → Set Availability.

[S2] Show weekly hours, scheduled breaks, order lead-time.

[S3] Edit open/close hours and holiday overrides, add special dates, and set delivery vs pickup windows. [A1]

[S4] Toggle “Pause Orders” (busy mode) or “Closed Now.”

[S5] Propagate status to customer app.

#### **Alternative Flows**

[A1] Invalid hour overlap: block save, suggest corrections.

## **UC7: Restaurant - New Order**

The restaurant receives, accepts, and begins preparing the order.

#### **Preconditions**

- UC3 - Order submitted and paid for.

#### **Main Flow**

1. System forwards order to restaurant POS/tablet; Status = “Confirming”. [S1]
2. The restaurant accepts the order and sets prep time; Status = “Preparing.” [S2]  
[A1]
3. Kitchen finishes preparation, marks order as complete; Status = “Ready” [S3]
4. Kitchen hands off order to driver; order is removed from active list [S4]

#### **Subflows**

[S1] Order is displayed on restaurant POS/tablet.

[S2] Restaurant confirms order, sets prep time. Order marked as “Preparing”.

[S3] Order marked as “Ready”.

[S4] Order marked as “On The Way”.

#### **Alternative Flows**

[A1] **No Response:** Notifications are resent to the restaurant. Order can be canceled/refunded by the customer after a prolonged no-response period.

## **UC8: Driver - Onboarding & Login**

A delivery driver joins the platform and is cleared to take jobs.

#### **Preconditions**

- Device with internet access.
- Driver App is open.

#### **Main Flow**

1. The driver taps the “Sign Up” button. [S1]
2. The driver provides basic login credentials [S2]

3. The driver provides their driver's license, background check consent, vehicle information, and banking details. **[S3] [A1]**
4. The driver submits documents. **[S4]**
5. The system runs checks and sets account status. **[S5] [A2]**
6. On approval, the driver logs in and sees the home screen. **[S6]**

#### **Subflows**

**[S1]** The driver selects "Sign Up".

**[S2]** Basic account information such as email and password are entered

**[S3]** Capture driver's license, insurance, plate, W-9/SSN (where applicable), bank/instant pay.

**[S4]** Upload photos and confirm signatures/consents.

**[S5]** Background/MVR checks; training acknowledgment.

**[S6]** Enable online/offline toggle and profile completion checklist.

#### **Alternative Flows**

**[A1] Bad Documents:** Documents are illegible or invalid; prompt re-upload.

**[A2] Checks failure:** Checks fail during the process - set "Hold" status. Allow uploading of new documents, opens support pathway.

## **UC9: Driver - Order Assignment**

System assigns a driver to pick up an order

#### **Preconditions**

- Orders are available

#### **Main Flow**

1. Driver goes online **[S1]**
2. Driver views list of orders available to pick up **[S2] [A1] [A2]**
3. A driver accepts an order and receives pickup instructions. **[S3]**

#### **Subflows**

**[S1]** Driver clicks the "Offline" button in the driver dashboard

**[S2]** Orders are listed based on total travel distance. Commissions are also shown.

**[S3]** Share pickup location, parking notes, and customer contact info.

#### **Alternative Flows**

**[A1] Batching:** Delivery jobs can be batched for improved efficiency

**[A2] No Driver Accepts:** Orders that have not been accepted for a while can get a bonus to incentivise pickup.

## **UC10: Driver - Pickup/Handoff**

Driver picks up an order from a restaurant and delivers it to the customer

#### **Preconditions**

- **UC9** - Driver has selected an order

#### **Main Flow**

1. Driver arrives at the restaurant and verifies order details. **[S1] [A1]**
2. Driver marks the order as picked up and departs. Status: "On the way". **[S2]**
3. Driver delivers the order to the customer. Status: "Delivered" **[S3]**

### **Subflows**

**[S1]** Driver verifies order code and bag count (and if it's sealed, if required).

**[S2]** System updates status to "On the way" and updates the ETA for the customer. Order is removed from the restaurant's tablet/POS.

**[S3]** System updates status to "Delivered" and the Driver can now accept new jobs.

### **Alternative Flows**

**[A1] Order Not Ready:** Driver arrival before readiness triggers wait policy.

## **Reflection Document**

### **How we decided what not to do**

When moving from the larger list of ideas in 1b1 to a minimal viable product, we had to make tough choices about what to leave out. The goal was to focus only on the features that prove the core journey works: finding restaurants, placing an order, having it prepared, and getting it delivered successfully. Anything that did not serve this direct purpose was set aside.

Some of the excluded features were attractive but would have demanded too much extra time or complexity. For example, we decided not to include global content moderation for reviews, what3words or Plus Codes for alternative addresses, or multilingual chat and escalation. These are helpful in the long run, but they are not necessary to show that customers can use the app to order food in our launch city. Similarly, loyalty and referral programs, sustainability reporting, or rider compensation policies were left out. While they may improve customer satisfaction or fairness for drivers, they do not directly teach us whether the basic ordering system will be adopted.

In a few cases, we also noticed that simpler solutions could achieve most of the same results. Instead of a full sustainability tracking system, we rely on menu tagging and allergen warnings to build trust. Instead of a driver resource portal, we focused on reliable onboarding and clean order assignment, which are essential for understanding supply and demand.

### **Negative impacts or disappointments this MVP could have**

Leaving out advanced features will naturally disappoint some groups. Customers might feel the app looks more basic than competitors because it does not include extras like loyalty points, multiple languages, or detailed sustainability dashboards. Drivers may

wish for clearer compensation policies or access to insurance and financial tools. Restaurants may miss having detailed analytics or fraud controls. Administrators will also need to work with lighter tools, since things like content moderation and broad reporting are not part of this first version.

Because of these gaps, some problems will still have to be solved through generic support channels. This could mean slower response times or less satisfying answers for unusual cases. It may also make the app feel less polished than what users are used to, especially for those who have experience with bigger platforms.

Despite these drawbacks, the decision to cut scope is important. The purpose of the MVP is not to match every feature that competitors already have, but to test the riskiest questions first. We need to know if customers can browse and check out without problems, if restaurants accept and prepare orders on time, if drivers reliably pick up and deliver, and if our support channels can handle the most common issues. Features that do not add to those answers can wait until later.

### **Changes we made to appease stakeholders**

Even with a narrow scope, we still chose to add a few features that help build trust with key groups. For customers, we included dietary restriction filters and allergen warnings to show that their health and safety are taken seriously. These are small additions, but they add real value. For restaurants, we made sure they can control menus, store hours, and order history. This gives them the flexibility they need to operate and lets us see how their decisions affect customer choices. For drivers, we focused on making onboarding smooth and order assignment reliable, so they can begin working quickly.

While compensation adjustments for wait times are not in place yet, we are already collecting the data that will let us design a fair system later. For administrators, we made sure that pricing and onboarding tools exist, even though detailed analytics and moderation are not yet part of the product.

In the end, the MVP is deliberately lean but still considerate of what matters most. Customers can browse, order, and pay. Restaurants can manage menus and availability. Drivers can join and take jobs. Administrators can set prices and manage access. This keeps development manageable while still giving every stakeholder enough value to take part. Most importantly, it lets us collect the data that will guide the next round of improvements. Instead of guessing, we will know what needs to be added because the first version will reveal where people struggle and what they ask for most often.

# Prompt History

## **This Project**

### **Ollama gemma3:12b, brainstorming basic minimal requirements**

You: List all of the potential routines in a basic food delivery app

Model:

Based on the provided documents, here's a list of potential routines in a basic food delivery app:

- \* **Registration/Login:** Customers can register and login [5, 6].
- \* **Restaurant Selection:** Providing a comprehensive restaurant selection [2].
- \* **Menu Browsing:** Browsing menus and selecting items [6].
- \* **Order Cart:** Reviewing, editing, and finalizing orders via a shopping cart [3].
- \* **Order Submission:** Calculating the total amount due, including taxes and delivery fees [3].
- \* **Real-time Delivery Tracking:** Tracking orders [5, 2].
- \* **Promotional Offers:** Displaying customizable food listings and promotional offers [3].
- \* **Customer Support:** Providing a sense of security and trust [5].
- \* **Order History:** Accessing order history [5].

You: Can you go more in depth, looking at the routines from the perspective of a restaurant, delivery driver, and customer?

Model:

Here's a breakdown of potential routines within a basic food delivery app, categorized by perspective (Restaurant, Driver, Customer), based on the provided context:

### **I. Restaurant Perspective**

- \* **Restaurant Onboarding:** Registration and setup within the platform [2].
- \* **Menu Management:** Adding, updating, and managing menu items [3, 6].
- \* **Order Receiving:** Receiving and confirming new orders [3, 4].
- \* **Order Preparation:** Preparing the order and marking it as ready for pickup [4].



- \* **Order Status Updates:** Updating order status (e.g., "Preparing," "Ready for Pickup") [4].
- \* **Performance Tracking:** Accessing data on sales, popular items, and customer feedback [2, 1].
- \* **Partnership Management:** Interacting with the platform regarding commission rates and promotional opportunities [2].

## II. Delivery Driver Perspective

- \* **Driver Onboarding:** Registration and verification process [2].
- \* **Availability Management:** Setting availability for deliveries [2].
- \* **Order Assignment:** Receiving and accepting delivery assignments [2].
- \* **Navigation & Route Optimization:** Using maps and GPS for efficient delivery routes [3, 4].
- \* **Order Pickup:** Picking up orders from restaurants [4].
- \* **Order Delivery:** Delivering orders to customers [2, 4].
- \* **Order Status Updates:** Updating order status (e.g., "En Route," "Delivered") [4].
- \* **Earnings Tracking:** Monitoring earnings and delivery history [2].
- \* **Communication:** Communicating with restaurants and customers regarding delivery issues [6].

## III. Customer Perspective

- \* **Registration/Login:** Creating an account or logging in [5, 6].
- \* **Restaurant Selection:** Browsing and selecting restaurants [2, 6].
- \* **Payment Processing:** Handling payment transactions [3].
- \* **Customer Support:** Contacting support for assistance [5].
- \* **Feedback/Reviews:** Providing feedback and ratings on restaurants and drivers [1, 2].
- \* **Profile Management:** Managing personal information and preferences [3].

You: Now also consider potential routines that the app administrators may need

Model:

Here's a breakdown of potential routines for app administrators, based on the provided context, in addition to the customer, driver, and restaurant routines previously listed:

## I. App Administrator Routines

- \* **User Management:** Managing customer, driver, and restaurant accounts (creation, deletion, suspension) [3, 5].
- \* **Restaurant Onboarding Management:** Approving/rejecting restaurant registrations and managing restaurant details [3].
- \* **Driver Onboarding Management:** Verifying and managing driver registrations [2].
- \* **Content Management:** Updating menus, promotional offers, FAQs, and other app content [3, 4, 5, 6].
- \* **System Monitoring:** Tracking app performance, security, and availability metrics [1, 5].
- \* **Security Management:** Monitoring for and responding to security threats, managing user data encryption [1].
- \* **Pricing Management:** Adjusting delivery fees and commission rates [3, 5].
- \* **Order Issue Resolution:** Handling escalated customer complaints and order problems (refunds) [4].
- \* **Reporting & Analytics:** Generating reports on app usage, sales, popular items, and customer feedback [1, 1].
- \* **Feature Updates & Deployment:** Managing and deploying new features and updates to the app [2, 6].
- \* **Payment Gateway Management:** Integrating and managing payment processing systems [4].
- \* **Database Management:** Maintaining and optimizing the app's database [3].

## Prompts used previously

### Iteration 1a1, Ollama Gemma 27b: (no chat history)

You are an expert requirements analyst. Using ONLY the provided CONTEXT, generate exactly 5 unique use cases for a FOOD DELIVERY APPLICATION in the precise format shown below. Your output MUST comply with every rule here.

STRICT FORMAT (match EXACTLY; no colons after headings; each item on its own line):

UC# <Title>

<one-sentence summary>

Preconditions

<precondition line 1>

<precondition line 2>

Main Flow

<step sentence ending with bracketed refs, e.g., ... [S1] [A1]>

<step sentence ... [S2]>

<step sentence ... [S3] [A2]>

Subflows

[S1] <subflow description for S1>

[S2] <subflow description for S2>

[S3] <subflow description for S3>

Alternative Flows

[A1] <alt description for A1>

[A2] <alt description for A2>

## HARD REQUIREMENTS

- Title line must start with: UC# <Title> (e.g., UC1 Browse Restaurants by Location).
- The second line is a single-sentence summary.
- Section headers are exactly: Preconditions, Main Flow, Subflows, Alternative Flows — on their own lines, with NO trailing colon.
- In Main Flow, EVERY operational step is a single line that ends with bracketed references to the subflows/alternatives it depends on, e.g., [S2] [A1]. Use [S#] for subflows and [A#] for alternative flows.
- In Subflows, define ONLY the [S#] identifiers that appear in that use case's Main Flow, one per line, prefixed with a space then the bracketed ID (e.g., " [S1] ...").
- In Alternative Flows, define ONLY the [A#] identifiers that appear in that use case's Main Flow, one per line, prefixed similarly.
- Numbering for S# and A# restarts within each use case: [S1], [S2], ... and [A1], [A2], ...
- Number the 5 new use cases UC1..UC5 (for THIS batch).
- NO bullets other than the bracketed S/A lines. NO extra headings. NO markdown. NO colons after section headers. Each sentence on its own line.

## UNIQUENESS & SCOPE

- Generate functionality specific to FOOD DELIVERY (customers, restaurants, riders/drivers, operations, support).
- Do NOT duplicate or substantially overlap the already covered use cases given in <<QUESTION>>.

## OUTPUT

Produce only the 5 formatted use cases, nothing else.

<<QUESTION>>

CONTEXT:

<<CONTEXT>>

**Zero-shot and careful prompting iteration 1b1, Chatgpt (no chat history)**

#### P1) Edge cases first

List 40 failure- and exception-oriented use cases across the entire lifecycle (browse→checkout→prep→handoff→post-order), including fraud, inventory drifts, ETA re-promises, reassignment, weather, no-go zones, and chargebacks. Prefer weird but realistic scenarios over basics.

#### P2) Logistics deep dive

Brainstorm 35 logistics use cases: batching, pick-sequence optimization, building/lobby access, geofenced handoff notes, mid-trip reassignment, cold-chain packaging, cross-docking hubs, customs pre-clearance, SPS certificates, and lost-shipment recovery.

#### P3) Payments/tax deep dive

Brainstorm 30 payments & tax use cases: multi-currency pricing, interchange optimization, 3-D Secure step-up, refund throttling vs. CSAT, chargeback representment/arbitration, VAT/IOSS filings, e-invoicing portals, negative balance/clawback, and revenue recognition.

#### P4) Ops/SRE & Privacy/Security

List 25 platform-ops/privacy/security use cases: incident playbooks, map/notification failover, disaster recovery, capacity/cost anomalies, API deprecation, access recertification,

PII redaction, DPIAs, breach notification, and model bias audits.

#### P5) Partners & B2B

List 20 partner/B2B use cases: POS/menu sync, 3PL handoffs, API key rotation, marketplace

SLAs, offboarding/data deletion, influencer/affiliate programs, and corporate invoicing/limits.

#### B) Careful Prompting: Trim scope & add 10 structured UCs (UC11–UC20)

Role: You are a senior software engineering tutor.

Task: Generate 10 additional Cockburn-style use cases for a LOCAL (single-city) food-delivery app that extend an existing set (UC1–UC10). Earlier UCs already cover: browse restaurants, build cart/customize items, promotions, checkout/payment, restaurant

acceptance & prep, item substitution approval, driver assignment & pickup, delivery & proof,

order issue/refund/redelivery, scheduled order. Do not duplicate those topics.

Output: Number them UC11–UC20. For each UC, include exactly these headings and labels:

- Primary Actor (Customer / Restaurant Manager/Owner / Restaurant Staff / Delivery Driver / Dispatch/Operations)

- Trigger (one line)
- Preconditions (bullets; cross-reference earlier UCs where relevant, e.g., “UC4 Checkout completed”)
- Main Flow (6–8 numbered steps, concise)
- Subflows / Extensions (S1, S2, ...)
- Alternative / Error Flows (AF1, AF2, ... and include a return point, e.g., “→ return to Main

Flow step 2”)

Scope rules (LOCAL ONLY): single city; no cross-border/drone/FX/customs. Include practical policy hooks (time windows, retries, photo/OTP evidence). Keep steps short.

Coverage candidates (pick 10 that are not in UC1–UC10): Address Book & Delivery Notes;

Building/Lobby Access & Pickup Points; Reorder & Favorites; Post-Delivery Tip Adjustment;

Share Live Tracking; Support Chat Triage (order-linked); Payment Method Management;

Privacy: Data Export & Deletion; Restaurant Hours & Holiday Overrides (manager); Courier

Reassignment Mid-Trip.

Quality gates: Match headings/labels exactly; add upstream UC references in Preconditions;

end each UC with a Postcondition.