# CSC 510 Project 1a1 — Food Delivery System: Problem Familiarization

**Group 20: Aanand Sreekumaran Nair Jayakumari, Sanjana Chandrashekar, Aditya Baradwaj Rajan**

## 1) Stakeholders

- Customers (end users placing orders)
- Restaurant Owners / Franchisees
- Restaurant Staff (manager, kitchen, packer)
- Delivery Partners / Couriers (employees or gig workers)
- Platform Admin / Operations
- Customer Support / Trust \ Safety
- Payments (payment processor, anti-fraud)
- Logistics/Maps (routing, ETA, geocoding)
- Marketing \ Ads (promotions, coupons)
- Finance \ Accounting (payouts, taxes)
- City/State Regulators (health dept., food safety, permits)
- Tax Authorities (sales tax, service fees)
- Third-Party Integrations (POS, printers, inventory)
- Data/Analytics (reporting, experimentation)

## 2) Stakeholder Biases Potential Clashes (five examples)

- Customers want low fees \ fast delivery; Couriers want fair compensation and safe, efficient routes. • Restaurants want longer prep windows to ensure quality; Customers want shortest possible ETA.
- Marketing wants aggressive promos; Finance wants sustainable unit economics and fraud control.
- Platform wants standardized menus for searchability; Restaurants want creative freedom \ complex customizations.
- Regulators require strict compliance (age-restricted items, health codes); Users expect frictionless checkout.

## 3) Prompt Crafting: Zero-shot vs. Careful Prompting

**Zero-shot prompting**. Give the LLM a broad instruction (e.g., 'list food delivery use cases') and accept the first output. Pros: fast, good for initial brainstorming. Cons: shallow coverage, missing edge cases, inconsistent structure.

**Careful prompting**. Constrain format and scope (actors, preconditions, main flow, subflows, alternatives), provide 1–2 in-domain examples, and require numbered steps and explicit assumptions. Pros: higher recall and quality, easier to grade and compare. Cons: slightly more effort to craft the prompt.

# 4) Use Cases (10)

## UC1 — Sign Up \ Verify Account

**Preconditions**

- User has a valid email or phone number; device connected to the internet.

**Main Flow**

1. User opens app/site and selects 'Sign up'.
2. System collects email/phone, name, password or SSO consent.
3. System sends verification code/link.
4. User verifies; system creates account and default profile.

**Subflows**

- Social sign-in (Google/Apple): system receives identity token and creates account.
- Device trust: remember this device for faster login.

**Alternative Flows**

- Invalid code/expired link → system allows resend.
- Existing account for identifier → system suggests login or account recovery.

## UC2 — Manage Addresses \ Delivery Preferences

**Preconditions**

- User is authenticated.

**Main Flow**

1. User opens 'Addresses' and taps 'Add new'.
2. System geocodes typed address or current location.
3. User saves address with labels (Home, Work) and instructions (gate code, leave at door).

**Subflows**

- Set default drop-off preferences (hand-off vs. doorstep).
- Accessibility notes (elevator, lobby pickup).

**Alternative Flows**

- Ambiguous address → system requests additional landmarks/map pin.
- Address outside service area → system explains coverage and waitlist.

## UC3 — Browse Restaurants \ Menus

**Preconditions**

- Delivery address is set.

**Main Flow**

1. User opens Home and receives a list filtered by address, hours, cuisine, rating.
2. User applies filters (price, dietary tags) and searches menu items.
3. System shows restaurant page with menu categories, item options, fees and ETA.

**Subflows**

- Availability by time window (breakfast-only items).
- Allergen \ dietary labels (gluten-free, nut-free).

**Alternative Flows**

- Restaurant temporarily offline or kitchen at capacity → system hides or shows 'busy' state.
- Menu item unavailable → system suggests substitutes.

## UC4 — Build Cart \ Customize Items

**Preconditions**

- Restaurant page is open.

**Main Flow**

1. User selects an item and chooses options (size, sides, doneness).
2. System updates price and nutrition; item added to cart.
3. User repeats for more items; system computes taxes, fees, and ETA.

**Subflows**

- Special instructions (e.g., 'no onions').
- Group orders: multiple users add items to shared cart.

**Alternative Flows**

- Option conflicts (e.g., both 'extra cheese' and 'no cheese') → system prompts to resolve.
- Cart minimum not met → system suggests add-ons.

## UC5 — Apply Promotions \ Fees Disclosure

**Preconditions**

- Cart contains at least one item.

**Main Flow**

1. User opens 'Promotions' and selects a valid coupon or automatic offer.
2. System recalculates subtotal, delivery fee, service fee, taxes, and tip preview.
3. User reviews transparent breakdown before checkout.

**Subflows**

- Location-based fees (tolls, city fees).

- First-order discounts or loyalty credits.

**Alternative Flows**

- Promo not applicable (merchant excluded, basket rules) → clear reason shown.
- Stacking conflict → system applies best single promo with rationale.

### UC6 — Checkout \ Payment

**Preconditions**

- User is authenticated; cart passes validation.

**Main Flow**

1. User selects delivery time (ASAP or scheduled).
2. User selects payment method (card, wallet, cash-less).
3. System authorizes payment and places the order; confirmation shown.

**Subflows**

- Save card on file (PCI-compliant tokenization).
- Split payment for group orders.

**Alternative Flows**

- Payment authorization fails → user retries or changes method.
- Scheduling window no longer available → system proposes nearest slot.

### UC7 — Restaurant Accepts \ Prepares Order

**Preconditions**

- Order is placed and routed to the restaurant.

**Main Flow**

1. System pushes order to restaurant tablet/POS.
2. Restaurant accepts; prep time estimate sent.
3. Kitchen prepares items; packer seals and marks ready.

**Subflows**

- Out-of-stock item → restaurant proposes replacement; user approves via app.
- Temperature-sensitive items flagged for insulated bag.

**Alternative Flows**

- Restaurant does not respond within SLA → order auto-cancels with refund.
- Prep delay → updated ETA pushed to user and courier.

### UC8 — Dispatch \ Courier Pickup

**Preconditions**

- Order is accepted; pickup window approaching.

**Main Flow**

1. System assigns a courier based on proximity, vehicle type, and performance.
2. Courier navigates to restaurant; verifies order code; picks up sealed bag.
3. System updates status to 'Picked up' and provides live ETA.

**Subflows**

- Stacked deliveries (multi-stop routing).
- Hot-bag compliance check.

**Alternative Flows**

- Courier cancels → system reassigns; ETA recalculated.
- Wrong package detected → courier/merchant escalate to support.

## UC9 — Delivery, Handoff \ Proof of Delivery

**Preconditions**

- Courier has picked up the order.

**Main Flow**

1. Courier follows route; user tracks on map.
2. Courier completes hand-off (in-person or doorstep per preference).
3. System captures proof (photo, code, or signature) and marks delivered.

**Subflows**

- Secure buildings (lobby pickup, call concierge).
- Contactless delivery photo requirement.

**Alternative Flows**

- Customer unreachable → courier waits per policy, then escalates to support.
- Incorrect address → courier/CS work to correct; if failed, return/dispose per policy.

## UC10 — Cancellations, Refunds \ Post-Order Support

**Preconditions** •

Order exists.

**Main Flow**

1. User opens 'Help' and selects an issue (missing item, cold food, delays).
2. System validates with telemetry (prep time, courier path, temperature sensors if available).
3. System offers compensation (refund/credit) or replacement; case is logged.

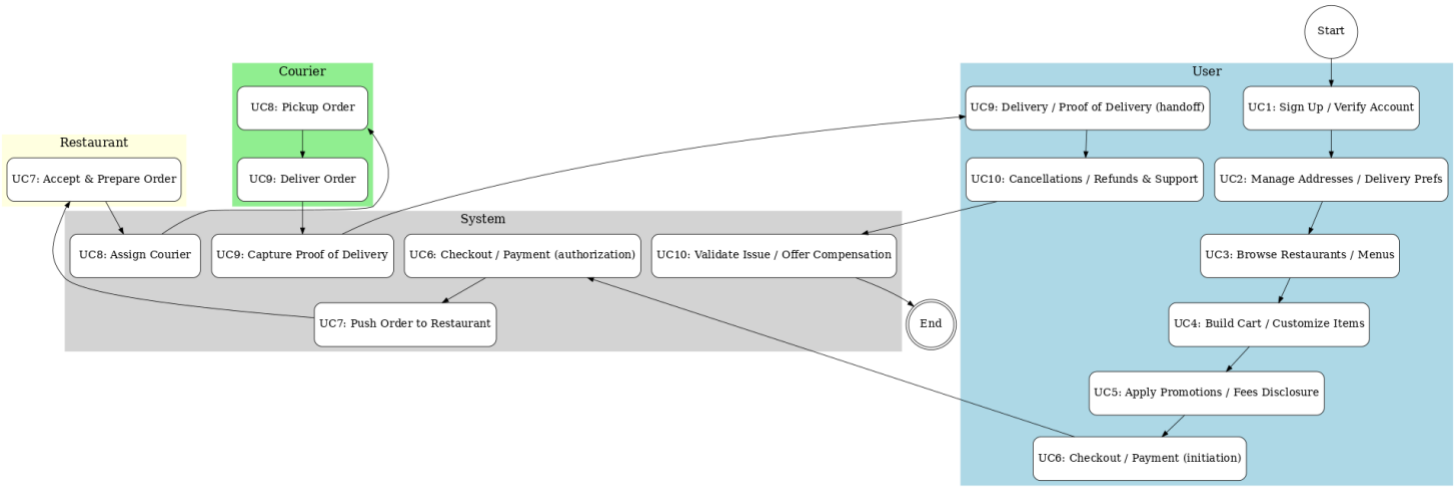
**Subflows**

- Partial refunds for specific line items.
- Quality assurance follow-up with restaurant/courier.

**Alternative Flows**

- Fraud signals (excessive claims) → route to manual review.

- Late cancel after prep → apply policy fees and notify stakeholders.




## 5) Appendix — Automatically Mining a Large Spreadsheet for Relevant Bits

- Export or read via Google Sheets API as CSV.
- Heuristic pass: filter rows by domain keywords (food safety, permits, sales tax, delivery zones, alcohol).
- Embedding pass: compute vector embeddings per row/section; query with tasks (*delivery fees disclosure*, *alcohol delivery compliance*).
- LLM pass: summarize top-ranked chunks into a compliance checklist with citations to cell ranges.
- Human validation: spot-check the top 20 items.

**Tooling sketch (Node/TS):** Use papaparse for CSV, compromise or wink-nlp for lightweight NLP, and cosine similarity over embeddings; emit a Markdown table of findings.

## 6) Submission Checklist Repo Layout

- Create a public GitHub repo (not NCSU) named 1a1-food-delivery.
- Add subdirectory proj1/ with this file as proj1/1a1.tex (or 1a1.md/1a1.pdf).
- Include a README.md summarizing stakeholders, method, and linking to the Google Sheet.
- Commit by the deadline; ensure the repo is public.

## References


- Cockburn, A. *Writing Effective Use Cases* (Addison–Wesley).

- IBM Docs — Use case specification outline.
- Wikipedia — Use case (definition and background).