<u>New Use Cases</u>

Use Case 1: Customer Creates an Account

Preconditions

- App downloaded, user not signed up.

Main Flow

1. Customer selects "Sign Up."
2. System shows form (email, name, password).
3. Customer submits info [Inline Validation].
4. System validates [Submission Error].
5. Account created, user logged in.

Subflows

- [Inline Validation] System highlights errors in real time (e.g., invalid email format, weak password strength, etc).

Alternative Flows

- [Submission Error] If the submission contains invalid or duplicate data, the system rejects the form and prompts the customer to correct it.

Use Case 2: Customer Logs In

Preconditions

- Customer has an account.
- Not logged in.

Main Flow

1. Customer selects "Log In."
2. System prompts for email + password.

3. Customer enters details [Recovery Option].
4. System verifies [Invalid Credentials].
5. Customer logged in.

Subflows

- [Recovery Option] "Forgot Password" link is shown directly below login fields.

Alternative Flows

- [Invalid Credentials] Show error if email or password is wrong.

Use Case 3: Save Multiple Delivery Addresses

Preconditions

- Customer logged in.
- In checkout flow or account settings.
- Location services available.

Main Flow

1. Customer goes to "Delivery Addresses."
2. System shows saved addresses [No Saved Addresses].
3. Customer adds new address (manual/GPS) [Address Validation Failed].
4. Customer labels and saves [Duplicate Address].
5. System stores address.

Subflows

- [Duplicate Address] Confirm saving similar address.

Alternative Flows

- [No Saved Addresses] Prompt to add first.

- [Address Validation Failed] Request correction.

Use Case 4: Restaurant Search with Filters

Preconditions

- Customer logged in.
- Location/address known.
- Restaurant data available.

Main Flow

1. Customer taps search bar.
2. Enters name, cuisine, or dish.
3. System displays results [No Results Found].
4. Customer applies filters [Apply Advanced Filters].
5. System updates results.
6. Customer selects restaurant.

Subflows

- [Apply Advanced Filters] Options: dietary needs, fees, promotions.

Alternative Flows

- [No Results Found] Suggest nearby or broader results.

Use Case 5: Promo Code Discovery & Application

Preconditions

- Customer has items in cart.
- Active promotions exist.

- Customer in checkout flow.

Main Flow

1. Customer proceeds to checkout.
2. System shows "Promos Available" [No Available Promos].
3. Customer selects promo or enters code [Invalid Promo Code].
4. System applies discount [Terms Not Met].
5. Checkout continues with updated price.

Subflows

- [Terms Not Met] Explain unmet requirements.

Alternative Flows

- [No Available Promos] Show standard price.
- [Invalid Promo Code] Show error message.

Use Case 6: Contactless Delivery Instructions

Preconditions

- Customer in checkout flow.
- Delivery address confirmed.
- Customer prefers contactless delivery.

Main Flow

1. Customer reaches delivery options.
2. System presents preferences [Special Instructions].
3. Customer selects "Contactless Delivery."

4. System prompts for instructions/photo [Photo Required].
5. Customer confirms.
6. Instructions sent to driver.

Subflows

- [Special Instructions] Notes (gate code, apt number).
- [Photo Required] Request photo proof.

Alternative Flows

- [Contact Required] Customer opts for in-person handoff.

Use Case 7: Estimate Delivery Time Before Order

Preconditions

- Customer viewing restaurant menu.
- Delivery address known.
- Prep/demand data available.

Main Flow

1. Customer views menu.
2. System shows estimated prep + delivery [Peak Hours Warning].
3. Customer adds items to cart.
4. System updates estimate [Extended Preparation Time].
5. Estimate confirmed before checkout.

Subflows

- [Peak Hours Warning] Show longer wait notice.

Alternative Flows

- [Extended Preparation Time] Complex orders extend prep time.

Use Case 8: Customer Places Order for Pickup

Preconditions

- Customer has items in cart.
- Pickup location and time selected.
- Valid payment method selected.

Main Flow

1. Customer reviews order summary.
2. Customer selects "Pay & Place Order."
3. System processes payment [Payment Successful].
4. System sends order details to restaurant [Order Received by Restaurant].
5. System displays confirmation with order ID and pickup time [Order Confirmation Display].

Subflows

- [Payment Successful] Transaction confirmed.
- [Order Received by Restaurant] Restaurant acknowledges new order.
- [Order Confirmation Display] Shows order ID, items, and pickup estimate.

Alternative Flows

- [Payment Failure] Prompt retry or alternative payment.
- [Restaurant System Unavailable] Notify customer, suggest retry.

Use Case 9: Order Modification Within Time Window

Preconditions

- Customer placed order 2–5 minutes ago.
- Status = Confirmed, not Preparing.
- Restaurant allows modifications.

Main Flow

1. Customer accesses active order.
2. System shows "Modify Order" with countdown [Modification Window Expired].
3. Customer edits items [Restaurant Confirmation Required].
4. System recalculates total.
5. Customer confirms.
6. System updates order [Payment Adjustment Failed].

Subflows

- [Restaurant Confirmation Required] Approval needed for major changes.

Alternative Flows

- [Modification Window Expired] Suggest contacting restaurant.
- [Payment Adjustment Failed] Revert to original order.

Use Case 10: Real-Time Order Tracking

Preconditions

- Customer has placed an order.
- Order status is "Confirmed" or later.
- GPS tracking enabled for driver.

Main Flow

1. Customer accesses "Track Order."
2. System shows order status (Preparing, Picked Up, En Route).
3. Driver location and ETA shown on map [Driver Location Unavailable].
4. Push notifications sent for updates [Delivery Completed].

Subflows

- [Delivery Completed] Prompt for rating/feedback.

Alternative Flows

- [Driver Location Unavailable] Show last known status.

Reflection

1) How Did We Decide What Not to Do

When we were putting together the MVP, one of the toughest challenges was deciding what not to build. Food delivery apps can get complicated fast if you try to pack in everything at once: things like loyalty programs, group ordering, split payments, gamification, or social features. Those are great ideas for later, but at the MVP stage, they would've added too much complexity and slowed us down.

For example, we skipped advanced group ordering. Coordinating payments and order limits across multiple people creates a lot of tricky edge cases that aren't worth solving right away. We also held off on loyalty programs and smart recommendations, since those need extra data collection and analytics, which don't bring much value in a first release.

In short, we avoided anything that required heavy backend logic, real-time sync between multiple users, or complex integrations. Instead we focused on the basics that give users immediate value: placing an order, tracking it, and managing their account.

2) Potential Negative Impacts or Disappointments for Stakeholders

Leaving out these “extra” features can sometimes disappoint different groups of stakeholders. Customers might find the app too basic compared to what they're used to, especially if they expect things like loyalty rewards, referral bonuses, or the ability to split the bill with friends. Restaurants could feel limited, too, since they miss out on upsell opportunities or insights that could help boost revenue. And for drivers, the absence of features like optimized routing or detailed delivery instructions might make their work a bit harder.

There's also a chance that some stakeholders see the MVP as unfinished or behind what's already available in the market. Early adopters might be let down if they don't immediately understand why this app is worth trying over existing options, which could mean losing potential users right from the start.

3) Changes Made to the MVP (and Why)

Even with these risks, we made small but meaningful adjustments to strike a balance. For example, while we didn't build a full loyalty or referral system, we did include a simple promo code flow. This lets us see how customers respond to discounts without creating a full rewards engine. Similarly, instead of removing delivery customization entirely, we added a lightweight contactless delivery option, showing that we prioritize both customer and driver safety and convenience.

We also added the ability to modify orders within a short time window, one of the most common pain points for customers and restaurants. While this adds a bit of complexity, it boosts customer confidence and reduces complaints from incorrect orders. These “middle ground” choices helped us keep the MVP lean while still addressing some key stakeholder needs.

In the end, the MVP isn’t perfect, and it won’t satisfy every stakeholder immediately. But by leaving out advanced or optional features, we can move faster and gather meaningful feedback. Everything we included was chosen to help customers order and receive food as smoothly as possible.

Prompt History

Claude Prompt History: https://claude.ai/share/7fdb0227-abd6-4801-86cd-9998d9cd8169
Gemini Prompt History: https://g.co/gemini/share/212deeafe2a8